

AFRL-IF-RS-TR-2002-15
Final Technical Report
February 2002



HIGH-PERFORMANCE INTEGRATION TRANSITION AND EXPLOITATION OF KNOWLEDGE (HITEK)

Teknowledge Federal System

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. F107

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-15 has been reviewed and is approved for publication.

APPROVED:

A handwritten signature in dark ink, appearing to read "Craig S. Anken".

CRAIG S. ANKEN
Project Engineer

FOR THE DIRECTOR:

A handwritten signature in dark ink, appearing to read "Michael Talbert".

MICHAEL TALBERT, Maj., USAF, Technical Advisor
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

| | | | | |
|--|---|--|---|----------------------------------|
| 1. AGENCY USE ONLY (Leave blank) | | 2. REPORT DATE FEBRUARY 2002 | 3. REPORT TYPE AND DATES COVERED Final Jun 97 - Sep 01 | |
| 4. TITLE AND SUBTITLE HIGH-PERFORMANCE INTEGRATION TRANSITION AND EXPLOITATION OF KNOWLEDGE (HITEK) | | | 5. FUNDING NUMBERS C - F30602-97-C-0145 PE - 62301E PR - IIST TA - 00 WU - 07 | |
| 6. AUTHOR(S) Adam Pease | | | | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Teknowledge Federal Systems 1810 Embarcadero Road Palo Alto California 94303 | | | 8. PERFORMING ORGANIZATION REPORT NUMBER N/A | |
| 9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency Air Force Research Laboratory/IFTD 3701 North Fairfax Drive 525 Brooks Road Arlington Virginia 22203-1714 Rome New York 13441-4505 | | | 10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2002-15 | |
| 11. SUPPLEMENTARY NOTES Air Force Research Laboratory Project Engineer: Craig S. Anken/IFTD/(315) 330-2074 | | | | |
| 12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. | | | | 12b. DISTRIBUTION CODE |
| 13. ABSTRACT (Maximum 200 Words) This effort focused on the development and integration of large knowledge bases and knowledge based tools. The project significantly advanced the state of the art in ontology and knowledge base development. Of particular scientific interest was a set of formal experiments that were conducted and described in technical reports and academic publications that quantified the value of knowledge base technology. The project also laid the groundwork for further efforts in ontology development and standardization. The Teknowledge integration approach was integration of components using a single large ontology based on Cyc. A significant result was the development of a version of Cyc for government distribution, which contained an Integrated Knowledge Base (IKB) and Integrated Development Environment (IDE). This software built on Cyc and included the ontology content developed during the HPKB project by our team. The HPKB project was structured to have several Challenge Problems that motivated and evaluate research progress. The Crisis Management Challenge Problem was focused on knowledge-based inference and the domain was reasoning about international crisis situations. The Battlespace Challenge Problem went through several iterations but wound up focusing on planning for battlefield workarounds and critiquing Army Courses of Action (COAs). The latter technology focus was on an integration of knowledge based sketching, restricted English to logic translation, problem solving, and knowledge-based inference. | | | | |
| 14. SUBJECT TERMS Knowledge Base Technology, Artificial Intelligence, Information Technology, Challenge Problem, Evaluation | | | | 15. NUMBER OF PAGES 37 |
| | | | | 16. PRICE CODE |
| 17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED | 20. LIMITATION OF ABSTRACT UL | |

TABLE OF CONTENTS

| | |
|--|----|
| INTRODUCTION | 1 |
| ACCOMPLISHMENTS IN 1997 | 1 |
| ACCOMPLISHMENTS IN 1998 | 2 |
| ACCOMPLISHMENTS IN 1999 | 3 |
| PRACTICAL KNOWLEDGE REPRESENTATION AND THE DARPA HIGH PERFORMANCE KNOWLEDGE BASES PROJECT | 8 |
| DOES PRIOR KNOWLEDGE FACILITATE THE DEVELOPMENT OF KNOWLEDGE- BASED SYSTEMS?..... | 16 |
| FROM VISUAL TO LOGICAL REPRESENTATION: A GIS-BASED SKETCHING TOOL FOR REASONING ABOUT PLANS | 22 |
| USING LARGE ONTOLOGIES TO ENABLE SEMANTIC INTEROPERABILITY OF PROBLEM SOLVERS | 28 |
| REFERENCES | 33 |

LIST OF FIGURES

| | |
|--|---|
| FIGURE 1 - CORRECTNESS OF COA CRITIQUING ANSWERS BY TOPIC AND NUMBER OF ANSWERS PER TEAM | 4 |
| FIGURE 2 - KNOWLEDGE BASE REUSE METRICS..... | 5 |
| FIGURE 3 - KNOWLEDGE BASE SIZE METRICS..... | 6 |
| FIGURE 4 - YEAR 1 CMCP PERFORMANCE | 7 |
| FIGURE 5 - YEAR 2 CMCP PERFORMANCE | 7 |

Introduction

The Teknowledge HPKB team integration effort resulted in the development and integration of large knowledge bases and knowledge based tools that were successfully transitioned to several DoD customers. The project significantly advanced the state of the art in ontology and knowledge base development resulting in a number of publications (Pease et al, 2001, 2000) (Cohen et al, 1998). Of particular scientific interest was a set of formal experiments that were conducted and described in technical reports and academic publications that quantified the value of knowledge base technology (Cohen et al, 1999). The project also laid the groundwork for further efforts in ontology development and standardization (Pease & Niles, 2002).

The Teknowledge integration approach was integration of components using a single large ontology based on Cyc. A significant result was the development of a version of Cyc for government distribution, which contained an Integrated Knowledge Base (IKB) and Integrated Development Environment (IDE). This software built on Cyc and included the ontology content developed during the HPKB project by our team.

The HPKB project was structured to have several Challenge Problems that motivated and evaluated research progress. The Crisis Management Challenge Problem was focused on knowledge-based inference and the domain was reasoning about international crisis situations. The Battlespace Challenge Problem went through several iterations but wound up focusing on planning for battlefield workarounds and critiquing Army Courses of Action (COAs). The latter technology focus was on an integration of knowledge based sketching, restricted English to logic translation, problem solving, and knowledge-based inference.

Key accomplishments of the integration team with respect to the challenge problems include:

- Delivery of a knowledge based reasoning system for the Crisis Management Challenge problem that consistently out-performed that of the competing integration team (see Figures 4 and 5 below)
- Delivery of an integrated battlefield plan critiquing system that included core technology for inference, plan sketching, and a large ontology created by our team. Our critiquing system answered all the challenge problem questions that the technology developers in the program did not want to address (which was roughly half the questions posed). See Figure 1 below.

A permanent record of the HPKB project and its results was created at <http://projects.teknowledge.com/HPKB>. To date there have been over a million unique accesses of this site which forms a significant resource for the AI community.

One programmatic conclusion that we reached during the course of the project was that the most successful integrations occur when the integration team also plays a role as technology developer, implementing research ideas as new components, as well as in providing “glue” for existing components.

In this report, we provide a narrative of accomplishments and a set of papers that describe the technical detail of some of those accomplishments.

Accomplishments in 1997

- Created Java interfaces for Cyc and NWU's SME (analogy reasoner). These interfaces were delivered very early in the course of HPKB and allowed us to make rapid progress later in interfacing systems.
- Early in the project we created the HPKB web site as a means for coordination and for recording progress. We posted kickoff meeting briefings to the web site at and worked with HPKB participants to define their integration deliverables and posted those deliverables to the HPKB web site. By the end of the first year of the project there were over 900 web pages on the site.
- We prepared a first version of an API document for our knowledge based system, a first version of the MELD language specification document that described the formal language that the system uses, and a first executable implementations including an Integrated Knowledge Base (IKB) and Integrated Development Environment (IDE).
- We reached agreement in the program to use a single upper ontology based on the Cyc upper ontology with additional contributions from Stanford and others.
- We gave several training courses on the Cyc system and ontology which were attended by most program participants
- We implemented an initial prototype GUI for the Crisis Management Challenge Problem (CMCP). This product allows a user to fill in information relevant to each phase CMCP problem solving. This system was presented to the CM SMEs and used to clarify information during initial Knowledge Acquisition (KA) sessions.

- We formalized and answered the Crisis Management Challenge Problem (CMCP) questions. Over half of the questions were answered successfully.
- We created a Network Flow Problem Solver (NFPS), based on (Ford & Fulkerson, 1956), integrated it with Cyc and developed a graphical front end. The interface allows a user to place transportation nodes on a map, define the traffic capacities between the nodes, and activate NFPS to find the bottlenecks in the system. This supported the Battlespace challenge problems. The integration with Cyc allowed us to reason with the output of the mathematical algorithm and make recommendations or answer questions about the traffic and trafficability situation on a battlefield.
- We integrated dynamic recognizers, developed by UMass, into our Situation Assessment problem solver. The integrated system reads data from a track file supplied by Alphatech, locates features in the data and asserts factual statements about the presence of those features to the ArcView GIS. This GIS-based graphical user interface supports visualization of battlefield events. It also includes some simple filtering rules that allow the large volume of data to be pared down to a reasonable size. Those features can then be sent on to Cyc for further knowledge based processing.
- We developed a Crisis Management Assistant based on a workflow specification created by AIAI. We also performed an initial integration with NWU's SME product.

Accomplishments in 1998

- We reviewed and assisted IET with their development of parameterized forms that can be used for specifying knowledge base queries and developed a GUI for entering those parameterized forms.
- We delivered a MELD-to-KIF translator
- We developed an ontology for battlefield workarounds
- Designed Workarounds problem solver, created specs for a planner and wrote axioms that the planner uses to solve the problem. Initial efforts included working with Alphatech to define the problem and working with MRJ personnel to define informally the rules that a workarounds engineer follows. Next, we developed the formal axioms in Cyc with support from AIAI. In addition, we worked with AIAI to integrate their workarounds problem solver with Cyc
- We created a translator which converts from the Alphatech Workaround input forms into CycL/MELD forms. This translator had the added benefit of discovering anomalies in the inputs, which were then communicated to Alphatech.
- We developed an interface between Java and the ArcView GIS in order to support the Workarounds Challenge problem.
- We delivered the full ontology required for stating the CMCP Parameterized Questions (PQs), instantiating them, and stating the answers to the instantiated PQ's.
- We created a data dictionary and then formalizations in logic of GIS data supplied by Alphatech. We then integrated this content with the Cyc KB.
- We began work on quantifying knowledge base size and rates of content development. (See figures 2 and 3 below).
- We added formal ontology content derived from EIA (Energy Dept) web pages, and CIA World Factbook. We developed new formal ontology content for preconditions and hypothesized actions and their consequences in support of the Workarounds Challenge Problem
- We integrated the NWU trafficability engine. This consisted of creating Cyc constants and axioms to allow communication of information, implementing a lightweight subset of the KQML protocol that NWU uses, and creating a GUI front-end.
- Our CMCP results were very good; they were high both in an absolute sense (generally in the 50-90% range) and relatively - surpassing the SAIC team on each and every one of the seven batches (see figure 4 below).
- Delivered a translator that converts from a vehicle movement database developed by Alphatech to CycL. The database, called FIRE&ISE, encodes all the input and output information for both the Workarounds and Movement Analysis challenge problems. By creating a product which could stand between the F&I database and the knowledge tools in HPKB, we avoided haggling over Alphatech's representation and allow the community to make progress on solving the challenge problem
- Completed integration of products from SRI, Stanford SMI, MIT, UMass. They were integrated through a monitoring system that filtered the input data according to user specified conditions and sent the filtered

data to the four systems. The monitoring systems then collected the results, converted them to Cyc and also sent the output to a visualization system built on the ArcView GIS

- Integrated the NWU trafficability engine. This consisted of creating Cyc constants and axioms to allow communication of information, implementing a lightweight subset of the KQML protocol that NWU uses and creating a GUI front-end. The software reasons about travel capabilities and travel times for various vehicles over terrain specified in a GIS.
- We delivered an initial ontology for the Battlespace COA problem. It includes over 100 constants and roughly five axioms for each constant and covers all the terms that Jim Donlon and Alphatech highlighted as being the most important in representing COA symbols. It includes a: planning ontology, military units ontology, high level military task ontology, specific military tasks, militarily significant areas, military purposes, task interaction relations, information sources ontology, and a Universal Transverse Mercator ontology. Successive versions of the ontology were posted on the HPKB web site in a version management system that allowed developers to track changes to the ontology.

Accomplishments in 1999

- We delivered a prototype COA drawing tool built on top of the ArcView GIS. This tool allows a user to place military unit symbols on a map and to draw phase lines and lines of responsibility. The information can be saved or loaded. A file interface format was also specified so that other tools can work with this information. The drawing tool generates formal logic statements that are equivalent to the visual content of the sketch.
- We worked with Textwise to integrate their text processing and CycL generation system with Cyc. This resulted in a system that took individual English sentences, converted them to Textwise's CRC representation and then translated a portion of these to CycL statements.
- We participated in the COA Critiquing CP evaluation, answering as many questions types as all the other participants combined. In addition, we generated the test problem input for ISI, GMU, and NWU in the COA ontology from a graphical representation and structured English test input. (see figure 1)
- We integrated of the NWU geographic reasoner with Cyc and used it in answering several of the CMCP test questions.
- We delivered a workarounds planning system that coupled plan generation rules to a faster planning algorithm than our initial Cyc-based approach. Several of the test problems can now be solved in a quarter of a second as opposed to several minutes with the old planner.

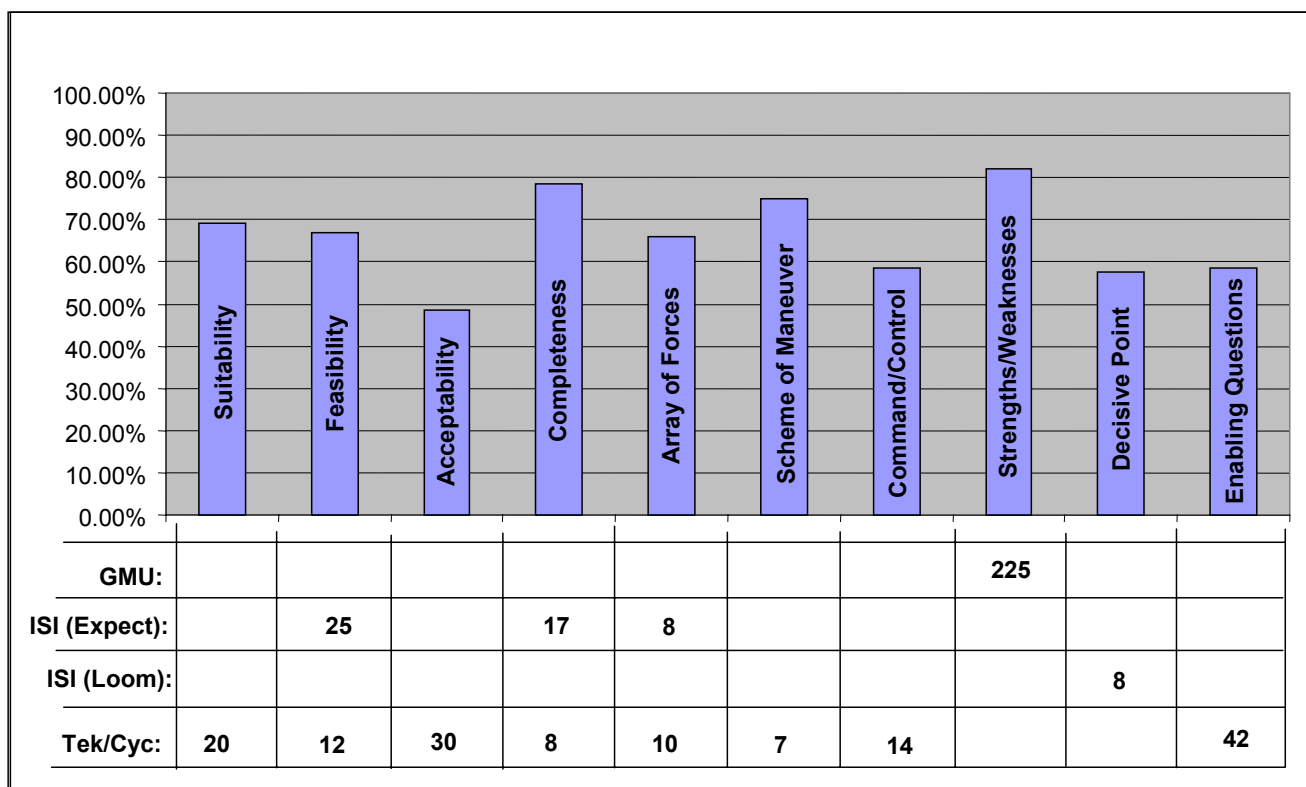


Figure 1 - Correctness of COA Critiquing Answers by topic and number of answers per team

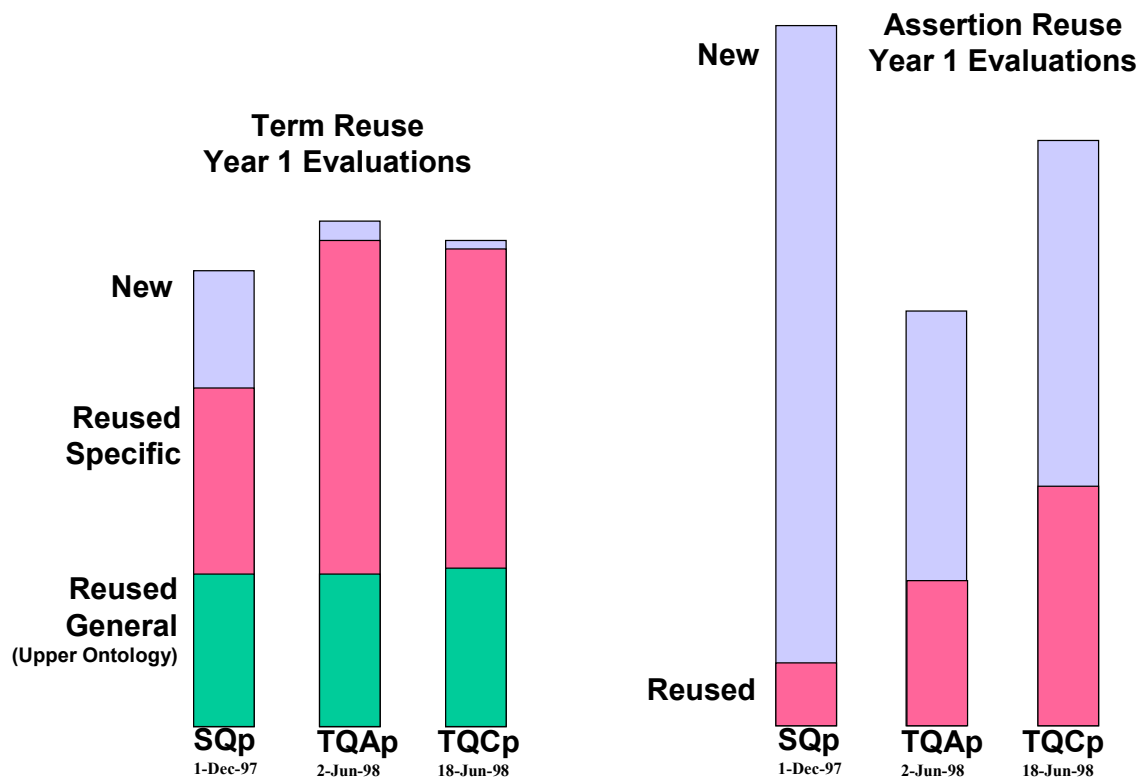


Figure 2 - Knowledge Base Reuse metrics

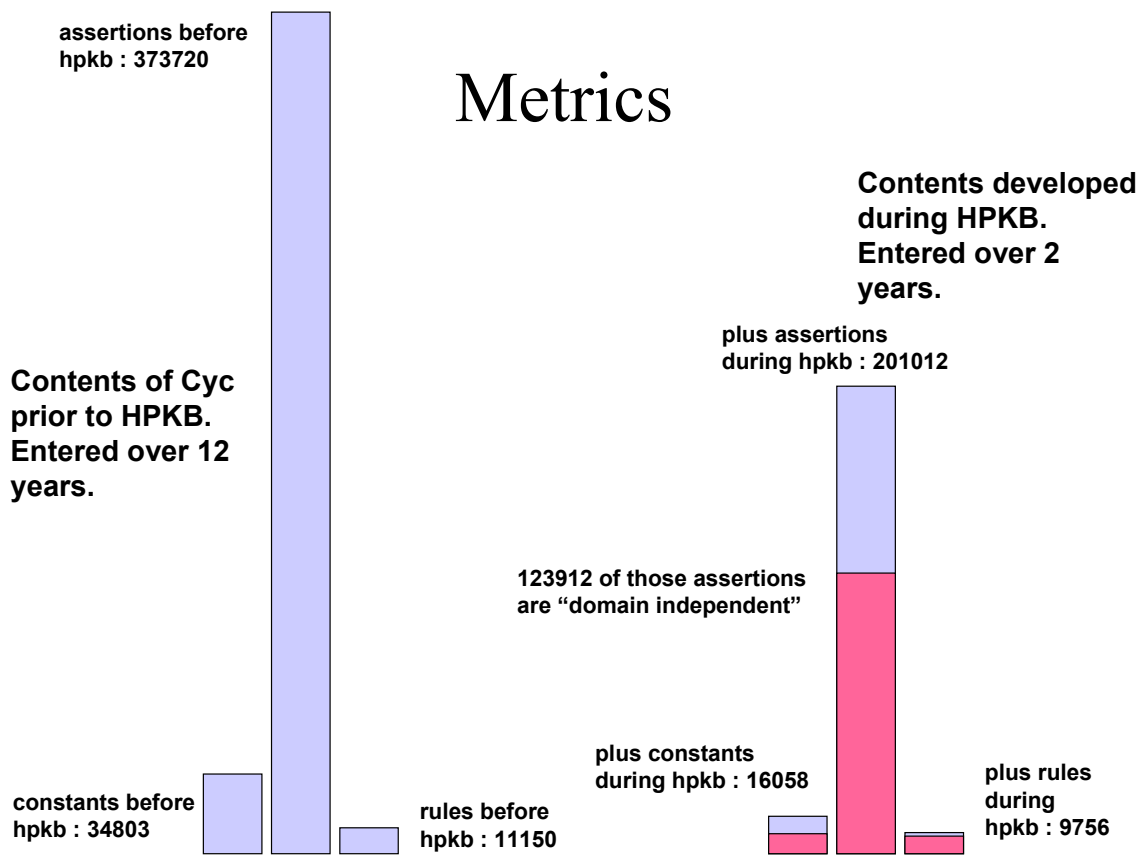


Figure 3 - Knowledge Base size metrics

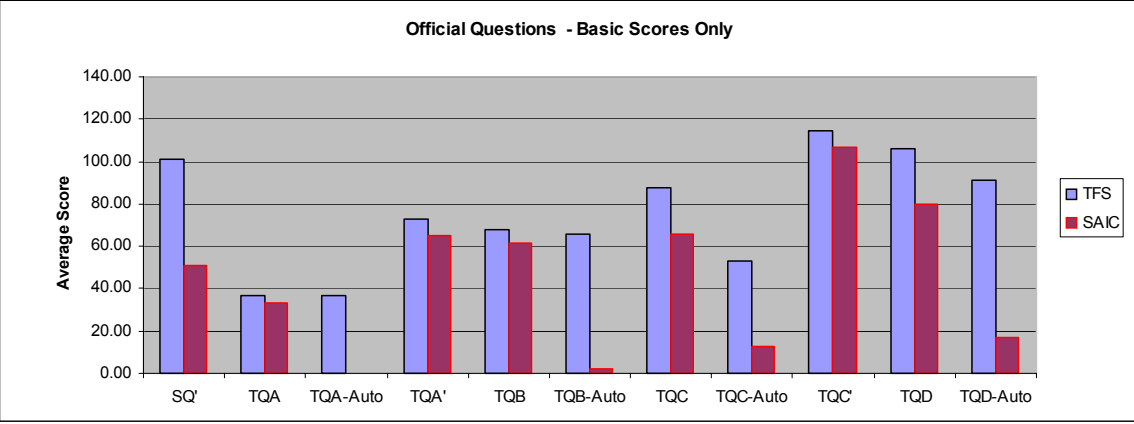


Figure 4 - Year 1 CMCP Performance

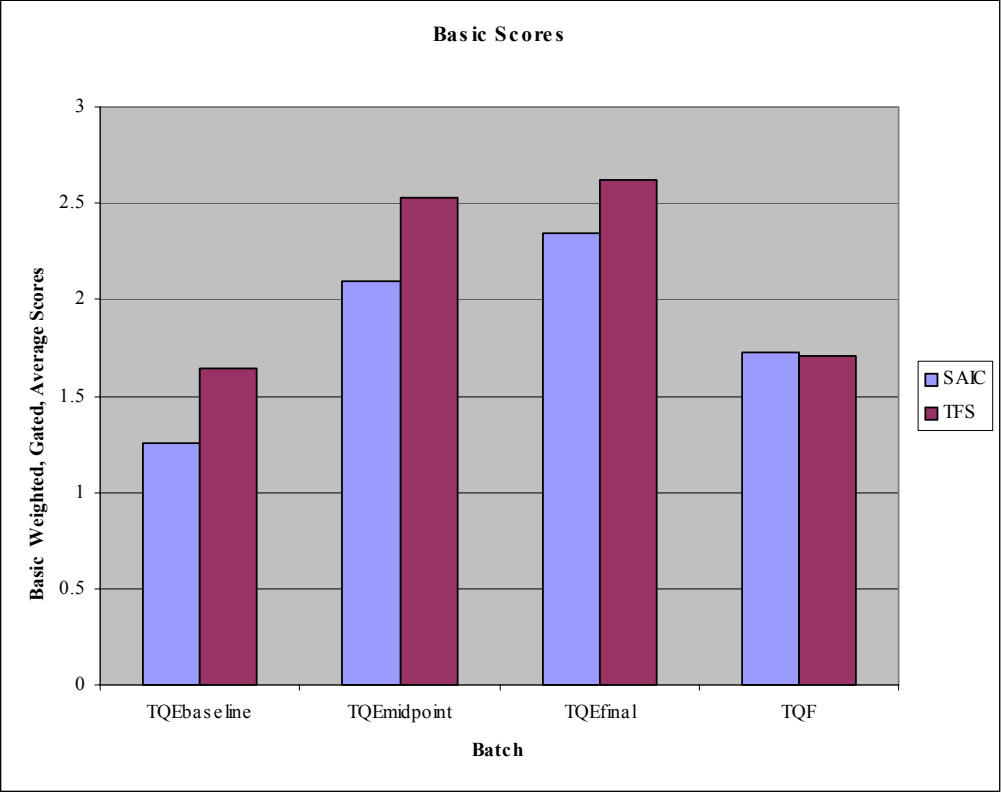


Figure 5 - Year 2 CMCP Performance

Practical Knowledge Representation and the DARPA High Performance Knowledge Bases Project

Adam Pease
Teknowledge
1810 Embarcadero
Palo Alto, CA 94303
USA
apease@teknowledge.com

Vinay Chaudhri
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
USA
chaudhri@ai.sri.com

Fritz Lehmann
Cycorp
3721 Executive Cntr Dr
Austin, TX 78731
USA
fritz@cyc.com

Adam Farquhar *
Schlumberger
8311 North FM 620 Road,
Austin TX 78726
USA
afarquhar@slb.com

Abstract

We address the experiences of the DARPA High Performance Knowledge Bases (HPKB) (Cohen et al., 1998) project in practical knowledge representation. The purpose of the HPKB project was to develop new techniques for rapid development of knowledge bases. The goal of this paper is to describe several technical issues that arose in creation of practical KB content.

HPKB PROJECT

EXPERIMENTS

The project had two main objectives: first, to advance the science of Artificial Intelligence Knowledge Representation and Knowledge Base content creation, and second, to apply these technologies to create applications with utility to the Department of Defense. The applications were specified as two Challenge Problems (CPs). The first was the Crisis Management CP, an effort to develop an automated question answering system that met the needs of analysts who must be informed about emerging world crises. The second was the Battlespace Challenge Problem. This effort covered two knowledge-based systems. One reasoned about battlefield engineering tasks such as workaround computation; the other critiqued battle plans. This paper addresses issues primarily from the experiences of the Crisis Management CP.

PROJECT ORGANIZATION

Two teams worked on these challenge problems. In the Crisis Management CP, one team used Cyc (Lenat, 1995) and its MELD (Cycorp, 1997) representation language. Another used KIF (Genesereth & Fikes, 1992) and the SNARK (Stickel et al., 1994) and ATP theorem provers.

HPKB was a very large project and many aspects are not even mentioned in this paper. The interested reader should refer to the HPKB web site (HPKB Web, 1999) and publications list (HPKB Pubs, 1999).

TRADEOFFS IN THEORY CREATION

There is a cost in creating reusable representations. It is more costly to create representations that will be reusable across multiple domains than it is to create a representation that is suitable for just one application.

We believe there is a need for a more formal development process that is built on some of the best practices from the software engineering community. It is always easier to create specific and limited content as opposed to crafting a general domain theory. The challenge is to build time into the development process for planning and systems analysis, design, implementation, testing, and *rework and generalization*. Much like the spiral development model advocated by Booch (Booch, 1994) and others, a good development process iterates through these stages several times during a development process. One possible instantiation of this process would be as follows:

DEVELOPMENT PROCESS

Planning and systems analysis. It is essential to determine the need that the knowledge must fulfill. Will it be used for inference? To define a semantics for natural language interpretation? As an interlingua for cooperating agents or software modules? Each of these

* The author performed this work while a member of the Knowledge Systems Laboratory, Stanford University

applications will entail a different emphasis on the richness of the formalization.

Also considered should be the performance requirements of the implementation. How fast should the resulting inference be? Will the knowledge base need to be augmented with a significant amount of instance data? Is logical completeness a necessity? Answering these questions will help to determine how expressive the knowledge representation can be, which will in turn partially determine the inference engine that needs to be employed.

We should note that in the HPKB project, a great deal of the systems analysis phase was done for the knowledge base developers by providing them with a Challenge Problem (Schrag, 1999:2) that specified and detailed the scope and purpose of the experiments that were to come. A great deal of informally specified knowledge was also provided.

Design. One way to design a knowledge base is initially to specify it informally. The engineer creates English examples illustrating sample reasoning chains. Glossaries with English definitions are created. It can also be useful to create a taxonomy as a skeleton on which the theory can be developed.

Implementation. As in software development, if the two previous phases are done properly, the implementation phase can proceed quickly. It is important that all members of the development team participate in the first two phases. Also helpful is a formal review process led by a chief knowledge architect.

Knowledge architects, software architects, and building architects all have similar roles. While they do not control every detail of a project, they set the overall design, standards, and aesthetics. A knowledge architect provides guidance to his team about how to meet project requirements, find a balance in tradeoffs between development speed and implementation generality, maintain consistent approaches across diverse team members, and set standards for reviews and documentation. A good architect manages by objectives and standards, which result in an implementation that speaks with one voice while allowing participants the freedom to innovate.

Testing. While this phase is obvious for any knowledge base that is to be used in a computational system, performing systematic testing is often ignored. If the knowledge base has been developed in a modular manner, an equivalent to *unit testing* can be performed on each small theory. Unit testing allows for testing of greater coverage than final *integration testing*.

Rework and Generalization. This phase is the most often ignored simply because of the dynamics of most research projects. Once the practical objectives of the sponsors have been achieved, little time or money

remains in the project to correct shortcuts that may have been made. However, this phase is possibly the most important if incremental scientific results are to be achieved.

Any large scale project will necessarily go through the above phases several times. A good knowledge engineering process has many similarities to a good software engineering process.

THEORY REUSE

Both teams reused the HPKB upper level (HPKB-UL) ontology, derived from Cyc, during the project. The representation for the temporal knowledge available in the HPKB upper ontology was very well designed. From the HPKB-UL, we also used representation for communicative actions, slots on actions (agent roles), and the primitives for representing paths. For one team, reusing these theories required translating the representation, extracting portions of the input ontology for use, and doing limited reformulation. There was also the need to further extend the library of the representation primitives for causality, scales, actions, processes, and qualitative influences.

The Cyc-based team had access to the entire Cyc knowledge base. In addition to areas mentioned for the upper level, there are good theories for concrete physical domains of all sorts. Theories of belief, goals, trust, and the expression of causality in nondeterministic human events are essential and less well developed.

HPKB had a good record of reusing terms and basic statements about terms. Developers gained a great deal of value from inheriting a large set of precise distinctions about things in the world, such as the differences among a goal, a plan, and a desire. However, comparatively little reuse of general rules was evident. This can be explained in several ways:

- It's hard to write truly general rules.
- Insufficient effort has been placed into writing general rules because of the pressures of day-to-day results.
- Practicalities of inference are such that a long chain of reasoning involving general rules doesn't work in a reasonable amount of time. One has to "short-circuit" the deep reasoning with special-purpose rules that make the inference tractable.

As an example of reuse, consider the following inference task performed by our system:

What risks can Iran expect in sponsoring a terrorist attack in Saudi Arabia?

To answer questions of this type, one team developed a simple cause-effect model. All the predicates below,

including *cause-event-event*, *beneficiary*, and *maleficiary* were reused from the HPKB-UL. Even though we capture only direct effects of an action, this simple model was effective in practice. This example illustrates the reuse of notions of causality that were already conceptualized in the HPKB-UL. The following is an example application of these representation primitives.

```
(forall ((?terrorist-attack
          terrorist-attack)
         (?agent agent))
(=>
  (performed-by ?terrorist-attack ?agent)
  (exists
    ((?punishment punishment))
    (and
      (causes-event-event
        ?terrorist-attack
        ?punishment)
      (maleficiary ?punishment ?agent)
      (object-acted-on
        ?punishment ?agent))))))

(forall ((?action action)
         (?action1 action1))
(implies
  (and
    (causes-event-event ?action ?action1)
    (performed-by ?action ?agent)
    (beneficiary ?action1 ?agent))
  (benefit-of-action
    ?action ?action1 ?agent)))
```

A detailed description of technical problems encountered in reuse is available in (Cohen et al., 1999) (Chaudhri et al., 2000). Even though we reused representations for actions and causality from HPKB-UL, significant additional representation work needed to be done. This suggests that a representation library for actions, causality, and qualitative influences needs to be extended. The theoretical KR community is invited to study the HPKB-UL and propose representational modules to be included in it.

PRACTICAL REPRESENTATIONAL ISSUES

There was a lack of principles for designing taxonomies. As a result, creating and maintaining a taxonomy of primitive concepts became increasingly difficult as its size grew. Conventional description logic techniques do not help in creating taxonomies that contain a large number of primitive concepts. Better principles for taxonomy design are needed.

There was also the need to "hand-compile" deep reasoning out into special-purpose theories that had tractable inference chains.

TAXONOMY

Like many other KBs, the class-subclass taxonomy was an overarching organizing principle in our HPKB KB.

A *class-subclass* taxonomy serves as an indexing aid to find knowledge and add new knowledge, and to serve as a method to efficiently write axioms by using inheritance.

While designing the taxonomies for the HPKB project, we encountered the following problems:

1. As the taxonomy got bigger, it became increasingly difficult to add new concepts to it. As a result, there were concepts that had incorrect positions in the taxonomy:

- Some concepts had missing links. A class has a missing super-class link if it is a subclass of another class B, but the subclass relationship is not declared.
- Some concepts had wrong links. A class has a wrong link in a taxonomy if it is a direct subclass of B, but the subclass relationship does not hold true.

2. We were encountering concepts that were being created by a cross-product of two sets of concepts, for example:

```
{International, transnational, subnational,
national} x {organization, agent}
{Support, oppose} x {attack, terrorist-
attack, chemical-attack} {Humanitarian,
political, military, diplomatic} x
{Organization, Action}
```

3. Some concepts had a very large number of subclasses. In some cases, this was due to orthogonal ways to categorize a concept. As a result, such categorizations were not mutually disjoint. Large fan-outs made it cumbersome to navigate through the taxonomy. As an example, consider the following snippet from the taxonomy representing organizations.



Figure 1. A portion of a taxonomy representing organizations showing orthogonal categorizations

While the categorization of commercial organization and unincorporated organization is based on the legal status of an organization, the categorization of international organization and subnational organization is based on extent of operations. Mixing such

orthogonal categorizations adds to the complexity of the taxonomy.

4. If two classes are disjoint, the disjointness relationship must be declared.

5. There should be no redundant classes representing identical concepts.

A taxonomy is well designed if it is free from all the problems mentioned above. Ensuring these properties in a small taxonomy is easy even if it is done manually. However, as the taxonomy size grows, making taxonomy well structured manually is very time consuming. These problems are indicative of a poor design methodology for developing taxonomies. We argue below that these problems go away if one takes a more principled approach to developing these taxonomies and supports additional constructs to structure the taxonomies.

If every concept has necessary and sufficient definitions, one can use a classifier to help alleviate Problem 1. In practice, we found that too many concepts were primitive and did not have necessary and sufficient definitions. Therefore, we cannot use a classifier. Problem 1 stems from the fact that the taxonomy itself is getting too complex. For example, a concept is linked or needs to be linked to too many different places. As a result, defining a new primitive concept involves manually encoding its relationship to numerous other primitive concepts -- a process that is error prone. One would hope that the process of organizing such concepts into a taxonomy would be considerably simpler than doing the same thing for the original concepts.

We need principles for taxonomy design that can enable us to economically create and maintain large taxonomies of primitive concepts.

COMPOSABLE REPRESENTATIONS

We believe that representations are more *reusable* if they are *compositionally* constructed. A representation is compositional if it represents each individual concept in the domain of discourse, and the representation of complex concepts is obtained by composing representations of individual concepts. To illustrate this, consider the representation of the following:

The USA conducts a peacekeeping mission.

In this example, we can use several different representations. One degenerate representation might be

UsConductOfPeacekeeping

This representation compiles all the semantic features of the English statement into a symbol. A more reasonable representation might be

```
(and
  (instance-of ?Y PeacekeepingOperation)
```

```
(performedBy ?X ?Y)
(members ?X USMilitaryOrganization))
```

in which the action has been expanded to describe an action type and detail about the performer of the action. We can further decompose the action by describing it as an event that has the purpose of maintaining a particular state.

```
(and
  (toMaintain ?Y PeaceAccord)
  (instance-of ?Y MilitaryOperation)
  (performedBy ?X ?Y)
  (members ?X USMilitaryOrganization))
```

(Schrag, 1999:1) has proposed the following compositionality hypothesis: noncompositional representations are inexpensive to build but they are brittle with respect to weak problem generalizations and must be re-engineered (for example, into compositional representations) or replaced.

According to the compositionality hypothesis, the first representation is inferior to the later versions. However, although many knowledge engineers would have a strong intuition that the later representations are superior, there is no strong empirical basis for the proposed criticism of the first representation. One approach that would admit the first representation as acceptable would be to add additional terms to the KB and give a more complete definition to it. Thus, even if the first representation is noncompositional, it is amenable to generalization if an application requires it.

The relative comparison between the two representations is unlikely to have a context-independent answer. If in the current application we never need to represent or reason with *conduct*, *mission*, or *peacekeeping*, other than talking about "conduct peacekeeping mission", the less expressive representation is adequate. One can certainly argue that the first representation is less reusable. However, that depends on the next application. If we use the first representation, and the next application requires us to represent or reason with *conduct*, *mission*, or *peacekeeping*, it is possible to add them to the KB and use them to define **UsConductOfPeacekeeping**. This may be studied more formally with an analytical model as follows.

Suppose we design two representations, one of which uses n_1 terms and the other uses n_2 terms. Suppose cost/term is c and is constant in both cases. The cost for building a KB for the two cases is $c*n_1$ and $c*n_2$, respectively.

If speeding up KB construction time for just one application is the objective, a compositional representation can be bad! However, if we also care about reuse, that may not be necessarily so. Does compositionality enable reuse? We cannot find out until we run replicated trials.

Suppose we reuse the KB for a new application. This new application requires the same knowledge fragment that we have already coded but requires a different compositionality, and we end up defining n_3 new terms for the first representation and n_4 new terms for the second representation. It is possible that either of n_3 or n_4 is zero. The cost for the new application is $c \cdot n_3$ and $c \cdot n_4$, respectively.

The objective should be to minimize $c \cdot (n_1 + n_3)$ or $c \cdot (n_2 + n_4)$. The model can be generalized to N applications. The parameter c can be viewed as time to construct a KB, and thus linked directly to the program goal of speeding up the KB construction time. Further, this model allows us to do the following:

- Measure whether it is really worth decomposing a representation
- Amortize the higher cost of decomposition over a number of applications
- Make explicit the relationship between reuse and compositionality

Exploring this tradeoff is open for future work.

"COMPILED" REPRESENTATIONS

One of the HPKB Challenge Problems dealt with reasoning about economic actions. One might encode the following chain:

```

There exist economic actions
which open markets -
  opening markets encourages
  exports -
    increasing exports improves
    a country's trade balance -
      positive trade balance
      improves economic health -
        all countries are
        interested in
        economic health

```

However, it may be that in practice, because of the complexity and compositionality of each of the encoded statements, and the depth of the inference, such a reasoning chain does not terminate in a reasonable amount of time. While an inference of depth five may not seem very taxing, consider the fact that this set of rules exists in a very large KB along with tens of thousands of others. The task of matching these particular rules and determining that huge numbers of others are irrelevant is time consuming.

The result is that to create a reasoning system that reaches a conclusion in a short amount of time, one might have to encode

```

There is a set of actions
which open markets -
  opening markets contributes
  to economic health -
    all countries are interested
    in economic health

```

along with defining a set of actions as subclasses of "opening markets" actions.

The goals of a project can strongly bias a knowledge engineer to the second representation. If a research team is scored, or a development team is paid on the basis of "correct" answers, compositionality and deep reasoning will be sacrificed.

METRICS

For any practical KB content creation work, there is a need to state crisply the competence level of a KB, and to make claims about increasing competence as the time goes along. Even though we know that there is an intuitive relationship between the size of a KB and its competence, there is no foolproof way functionally to relate the size to competence. As an approximate measure, we used the axiom count in a KB as one measure of competence.

An early challenge during the project was to define what counts as an axiom. Given that there is no universal way to count axioms, and that the axiom counts are sensitive to the modeling style and the language, we developed the following scheme for categorization of axioms in a KB.

- Constants** are any names in the KB, whether an individual, class, relation, function, or a KB module
- Structural statements** are ground statements using any of (Cyc term/Ontolingua term)
 - `#$isa/instance-of`, `#$genls/subclass-of`,
 - `#$genlPreds/subrelation-of`,
 - `#$disjointWith/disjoint`,
 - `#$partitionedInto/disjoint-decomposition`,
 - `#$thePartition/partition`, `#$genlMt`,
 - `#$argXIsa/nth-domain` (where X is a digit),
 - `#$argXgenls/nth-domain-subclass-of` (where X is a digit),
 - `#$arity/function-arity/relation-arity`, `#$resultIsa/range`,
 - `#$resultGenls/range-subclass-of`
- Ground facts** are any statement without a variable.
- Implications** include any non-ground statement that has an `#$implies` (note that a ground statement that contains an `#$implies` is counted as a ground statement)
- Non-ground, non-implications** are statements that contain variables but not an implication.

This categorization is imperfect, but it is easy to implement and was applicable to both of the crisis management systems developed during the HPKB project.

The structural statements have an intuitive status in most systems: for SNARK the structural information is sort information, for Cyc the structural information is

called definitional, and for description logic systems the structural relations are usually called *concept constructors*. The statements with implications are rules. Ground facts often represent knowledge that can be found in an almanac or database.

A weakness of this categorization is that it counts many statements as ground statements even though they are not actually ground. For example, the statements involving `template-slot-value`, and `#$relationAllExists` are counted as ground. Further refinement to this categorization is left open for future work.

The axiom categorization scheme gave us an empirical tool to compare content across the two systems developed in the project. We would welcome proposals from the theoretical KR community, detailing more systematic ways to measure the competence of a large KB.

STANDARDS

Having a standard syntax is a necessity, but standard syntax plays a relatively small role in addressing the practical challenges facing the knowledge engineer. There is a need to move from an emphasis on standards of syntax, or on defining a precise semantics for tiny theories, to standard large theories and style guides for axiom writing.

For example, the subclass relationship can be either stated as

1. *(subclass-of A B)*, or as
2. *(=> (A ?x) (B ?x))*

Both of these forms are ANSI KIF. The first form uses *subclass-of* as a relation to compactly encode information that could also be written as in Form 2. The first form also has the advantage that a reasoner supporting taxonomic inference can take advantage of this form, which can be quite difficult for the second form.

As another example, consider three commonly used ways to specify the type information of variables in an axiom: (1) using ANSI KIF-style typed quantifiers, (2) using *instance-of* relations, or (3) using the class as a relation. Here is an example axiom encoded in these three forms:

```
(forall ((?x action)
        (?y action)
        (?z country))
  (=>
    (and
      (may-cause ?x ?y)
      (performed-by ?x ?z)
      (maleficiary ?y ?z))
      (risk-of-action
        ?x ?z ?y)))

(forall (?x ?y ?z)
  (=>
    (and
      (instance-of ?x action)
      (instance-of ?y action)
      (instance-of ?z country)
      (may-cause ?x ?y)
      (performed-by ?x ?z)
      (maleficiary ?y ?z))
      (risk-of-action ?x ?z ?y)))

(forall (?x ?y ?z)
  (=>
    (and
      (action ?x)
      (action ?y)
      (country ?z)
      (may-cause ?x ?y)
      (performed-by ?x ?z)
      (maleficiary ?y ?z))
      (risk-of-action ?x ?z ?y)))
```

One additional factor might be that may-cause and risk-of-action could be defined as referring to types of actions rather than instances in different knowledge bases.

These three forms are equivalent and follow the ANSI KIF standard. In spite of the standard, people come up with sufficiently different ways to write axioms to make the knowledge exchange difficult. Therefore, the standards must be accompanied by a style guide before they can enable knowledge exchange. In the above example, the style guide could require that the type information for axioms should always be stated in the quantifier specification.

USING A VERY EXPRESSIVE REPRESENTATION

Expressive representations enable a degree of generality and reuse not possible with more restricted representations. Because of interactions among axioms, the inference time can become very high. The most general and reusable theory is not useful if inference on those theories is not tractable for your inference engine. Some ways of addressing this problem are by partitioning the KB into modules to isolate the interactions among axioms, and by

compiling knowledge by hand into more efficient representations.

One team had the goal of keeping the inference time for answering a question to less than 2 minutes. If all the axioms were loaded at the same search space, it was not possible to meet this requirement. Therefore, we *modularized* the KB to limit the interactions among axioms and achieve the desired response time. This problem would have been less critical had we limited the representation to horn clauses.

KB modularization means dividing the content of a KB into conceptual partitions that serve the basis for KB development and inference. We experimented with two ways to modularize a KB: subject based and task based. A *subject-based modularization* organizes a KB by subject area and can enable easier sharing and development of KB content. A subject area can be assigned to a knowledge engineer to direct its development. While reusing a KB, one can select a KB in the subject area of interest. A *task-based modularization* organizes a KB by the rules and individuals that are relevant to a task, thus significantly reducing the search space. The class, function, and relation definitions do not affect the search space, and therefore need not be modularized to speed up inference.

Modularization of a KB based on the subject-based criteria and the task-based criteria can be different and can coexist. We used both subject-based and task-based modularization during the project. For example, three major subject areas covered in our KB are *actions*, *agents*, and *interests*. We also created task-specific partitions in the KB based on specific parameterized questions (PQs). For example, for answering questions about interaction between interests and actions, there was no need for knowledge about specific terrorist groups in the KB that were kept in a separate partition. The approach to modularization described here was clearly engineering driven, and better principles to arrive at the modularization are needed. Techniques to develop modules for a KB in a way that isolates independent reasoning chains are clearly of special importance.

ISSUES IMPEDING PROGRESS

Inference engine performance is one crucial technical issue. While it is not easy to develop inference modules for very expressive features, it is incredibly hard to get those modules to perform well.

Despite the program's name, execution speed was not an issue under investigation in HPKB. Many researchers have studied algorithms, speed, and complexity. HPKB was extremely important because it focused on content. Much research on inference

performance has not been undertaken in the context of practical reasoning on large knowledge bases. The challenge now is to focus on merging research on creating and reasoning with large knowledge bases with research on inference performance.

The most important nontechnical issue is research parochialism. The need to "own" a language, ontology, theory, or protocol is very powerful, whether in terms of building a research identity or a commercial base. However, this fragmentation is hampering progress.

Allen's seminal work (Allen, 1984) (Allen, 1994) on representing temporal knowledge is a good example of the kind of results that we need, and it is also well referenced and adopted in the applied AI community. Allen's work identified the primitives necessary to represent a sufficiently large class of temporal information and proposed inference procedures. If we could do the same for other domains such as actions, space, and causality, etc, it would greatly speed the practical KB construction. It is also the case that careful theoretical work has been done in these areas but may not be well known or adopted in the applied AI community. This work includes (Cohn et al., 1997), (Giunchiglia & Lifschitz, 1998), (Giunchiglia & Lifschitz, 1999), (Lifschitz, 1987), (McCain & Turner, 1997).

The KR community is still theoretically focused. Few people are interested in working on creating KB content. The time is right for a new focus on practical KB content creation.

Acknowledgments

We wish to acknowledge our DARPA sponsor, Murray Burke, for funding and guiding this work. We also wish to acknowledge the essential contribution of Robert Schrag at IET, who specified the Challenge Problem that made this research possible. Cleo Condoravdi provided a very helpful review of the paper.

References

- Allen, J. (1984). "Towards a General Theory of Action and Time", *Artificial Intelligence* 23, pp 123-154.
- Allen, James and George Ferguson (1994). "Actions and Events in Interval Temporal Logic", *Journal of Logic and Computation* 4, 531-579.
- Booch, G. (1994). *Object-Oriented Analysis and Design With Applications*, Addison-Wesley
- Chaudhri, V., J. Lowrance, J. Thomere, M. Stickel, and R. Waldinger (2000).

- Ontology Construction Toolkit. Artificial Intelligence Center, Technical Report.
- Cohen, P., V. Chaudhri, A. Pease, and R. Schrag (1999). "Does Prior Knowledge Facilitate the Development of Knowledge Based Systems", Proceedings of AAAI-99.
- Cohen, P., R. Schrag, Jones, A. Pease, Lin, Starr, Gunning, and Burke (1998). "The DARPA High Performance Knowledge Bases Project", AI Magazine, Vol. 19 No.4, Winter.
- Cohn, A., B. Bennet, J. Gooday, and N. Gotts (1997). Representation and Reasoning with Qualitative Spatial Relations about Regions.
http://www.scs.leeds.ac.uk/spacenet/leeds_qsr.html
- Cycorp (1998). "Features of the CycL Language", on-line report at <http://www.cyc.com/cycl.html>.
- Genesereth, M., and R. Fikes (Editors) (1992). Knowledge Interchange Format, Version 3.0 Reference Manual, Computer Science Department, Stanford University, Technical Report Logic-92-1, June.
- Giunchiglia, E., and V. Lifschitz (1998). An action language based on causal explanation: preliminary report. In Proceedings AAAI-98, pp. 623-630.
- Giunchiglia, E., and V. Lifschitz (1999). "Action Languages, Temporal Action Logics and the Situation Calculus". In Working Notes of the IJCAI-99 Workshop on Nonmonotonic Reasoning, Action, and Change.
- HPKB Web (1999). "HPKB Web Site", <http://projects.teknowledge.com/HPKB/>
- HPKB Pubs (1999). "HPKB Publications Page", <http://projects.teknowledge.com/HPKB/Publications.html>
- Lenat, D., 1995, "Cyc: A Large-Scale Investment in Knowledge Infrastructure". Communications of the ACM 38, no. 11, November.
- Lifschitz, V. (1987). "Formal Theories of Action". The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop. Los Altos, CA: Morgan Kaufmann Publishers.
- McCain and Turner (1997), "Causal Theories of Action and Change", Proceedings of AAAI-97, pp 460-465.
- Schrag, R. (1999:1), email communication.
- Schrag, R. (1999:2). "HPKB Year 2 Crisis Management, End-to-end Challenge Problem Specification", Version 1.2, February 5, Information Extraction and Transport, Inc. and Pacific-Sierra Research Corp. Rosslyn, VA.
<http://www.iet.com/Projects/HPKB/Y2/Y2-CM-CP.doc>
- Stickel, M., R. Waldinger, M. Lowry, T. Pressburger, and I. Underwood (1994). "Deductive Composition of Astronomical Software from Subroutine Libraries", in Proceedings of the Twelfth International Conference on Automated Deduction (CADE-12), June, 341-355.

Does Prior Knowledge Facilitate the Development of Knowledge-based Systems?

Paul Cohen,
Cohen@cs.umass.edu
U. of Massachusetts

Vinay Chaudhri,
chaudhri@ai.sri.com
SRI International

Adam Pease,
apeace@teknowledge.com
Teknowledge Inc.

Robert Schrag
schrag@dc.iet.com
IET Inc.

Abstract

One factor that affects the rate of knowledge base construction is the availability and reuse of *prior knowledge* in ontologies and domain-specific knowledge bases. This paper reports an empirical study of reuse performed in the first year of the High Performance Knowledge Bases (HPKB) initiative. The study shows that some kinds of prior knowledge help more than others, and that several factors affect how much use is made of the knowledge.

Introduction

With current technology, trained knowledge engineers can build knowledge bases at a rate of roughly 10,000 axioms per person per year, or roughly five axioms/person/hour. One factor that affects this rate is the availability and reuse of *prior knowledge* in ontologies and domain-specific knowledge bases. Until now, there have been no systematic studies of knowledge reuse. This paper reports an empirical study of reuse. The study was performed in the first year of the High Performance Knowledge Bases (HPKB) initiative sponsored by the Defense Advanced Research Projects Agency (Cohen et al., 1998). By comparing the efforts of two HPKB groups under different conditions, we find that prior knowledge in the form of ontologies does help, though many factors affect how much it helps. This work also introduces metrics and methods for evaluating the contribution of prior knowledge to knowledge-based systems.

By *prior knowledge* we mean the knowledge one has available in an ontology or knowledge base prior to developing a knowledge-based system. Several large ontologies have been developed including Cyc (citation) LOOM (citation), <Bruce Porter's system>, ..., ¹. All these systems contain hierarchies of knowledge. At the upper levels, one finds knowledge that is general to many applications, such as knowledge about movement, animate agents, space, causality, mental states, and so on. The lower levels contain knowledge specific to domains; for example, rules for inferring the effects of tactical military operations. Bridging general and specific knowledge, one finds *micro theories* (citation); collections of terms and

axioms about phenomena such as human physiology, more general than a particular medical expert system but less general than, say, knowledge about physical systems. In addition to hierarchies of terms, all the ontologies cited above contain *axioms*, for example, "all universities are educational institutions"; *rules*, for instance, "if x is an educational institution then x pays no taxes"; and inference methods such as resolution or more specialized forms of theorem-proving. Axioms and rules confer a functional kind of *meaning* on the terms they contain, that is, the meaning of a term is the things one can legitimately say (infer) about it.

One claim of ontologists is that it is easier to build a domain-specific knowledge base **KB** inside an ontology **O**, or informed by **O**, than without **O**. Some of the ways that **O** can help are illustrated in Figure 1. First, a term **p** that you wish to add to **KB** might already exist in **O**, saving you the trouble of adding it. Second, axioms relating to **p** might already exist in **O**, saving you the trouble of thinking them up and encoding them. Third, within **O**, **p** might be a subclass of **v**, so you also have the benefit of axioms about **v** inherited through **p**.

Now suppose you want to add a concept **p'** to **KB**, and **p'** is not exactly **p**, but is similar in some respects. For instance, **p** might be part of a microtheory about economics, and **p'** might belong to a microtheory about fluid flows, but both **p** and **p'** represent the concept "source." More generally, suppose the *structure* of the theory of economics in **O** parallels the structure of the theory of fluids that you are trying to build in **KB**. Thus, a fourth way that **O** can help you to build **KB** is to help you structure the theory in **KB**. Designing the structure of microtheories is very time consuming, so this kind of help may be the most important of all.

¹ See also <http://...> for a web site devoted to ontology-building efforts.

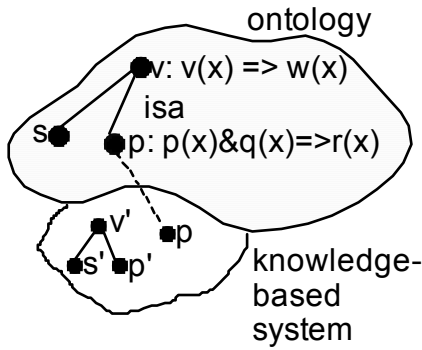


Figure 1. Some ways an ontology O can help one build a knowledge base KB .

Unfortunately it is difficult to assess experimentally how the structure of O helps one build KB s with similar structure, so we focus here on the first three ways that O can help one build KB .

Metrics

Suppose one wishes to add an axiom, “If x is a state then x maintains an army,” to KB . This axiom contains three terms, **state**, **maintains**, and **army**. Suppose the first two terms already exists in O but **army** does not. As two thirds of the terms required to add the axiom to KB exist in O , we say the *support* provided by O in this case is $2/3$. In general, every item i one wishes to add to KB contains $n(i)$ terms, $k(i)$ of which are already in O , and support is $s(i)=k(i)/n(i)$. Of course, adding **army** to O changes O , and the support offered by O for future axioms might be higher because **army** was added. Therefore, support is indexed by versions of the ontology: $s(i,j)=k(i,j)/n(i)$ is the support provided by version O_j of the ontology for concept i .

Experiment Design

We evaluated the support provided by ontologies during a month-long process called the *Crisis Management Challenge Problem* (CMCP). The CMCP was designed by Bob Schrag and his colleagues at IET, Inc. and PSR Corp. Two integrated knowledge-based systems were developed to answer questions about international crises, such as, “What will the US response be if Iran closes the Strait of Hormuz?” (Cohen et al., 1998). The systems were developed by Teknowledge and SAIC. The CMCP had several phases:

1. Some months before any testing began, a *crisis scenario* was released. The scenario bounded the domain and thus the scope of the problems to be solved by the Teknowledge and SAIC systems.

2. Several weeks before testing, a batch of sample questions (SQs) was released.
3. On the first day of the evaluation, a batch of 110 test questions, TQA, was released, and the Teknowledge and SAIC systems were immediately tested. After four days for improvements, the systems were re-tested on TQA.
4. Batch TQB was released immediately after the retest. The purpose of TQB, which contained questions similar to those in TQA, was to check the generality of the improvements made to the systems.
5. After a brief respite, a change was made to the crisis scenario, increasing the scope of the problems that the Teknowledge and SAIC systems would have to solve. Several days were allowed for knowledge entry prior to the release of a new batch of questions, TQC, reflecting the new scope. The systems were tested immediately.
6. Four days were allowed to extend the systems to the new crisis scenario, then the systems were re-tested on TQC. To check the generality of these extensions, the systems were also tested on batch TQD, which was similar to TQC.

One of the methodological innovations of this experiment was to generate all the batches of questions from a *question grammar* – a set of parameterized questions – which had been made available to the participants in the experiment several months before testing began. Batches SQ, TQA and TQB were generated by one grammar. The grammar was extended to reflect the change in the crisis scenario and used to generate batches TQC and TQD. Figure 2 shows one of the parameterized questions (PQ53) from the grammar. Many billions of questions could be generated by the question grammar, so it would not have made sense to develop systems to solve particular questions; however, by getting the PQs early, the system developers could limit the scope of their systems to the subjects mentioned in the PQs (e.g., terrorist attacks, EconomicSector, etc.)

PQ53 [During/After <TimeInterval>.] what {risks, rewards} would <InternationalAgent> face in <InternationalActionType>?

<InternationalActionType> =

{[exposure of its] {supporting,
sponsoring} <InternationalAgentType in
<InternationalAgent2>, successful terrorist attacks
against <InternationalAgent2>'s <EconomicSector>,
<InternationalActionType>, taking hostage citizens of
<InternationalAgent2>, attacking targets

<SpatialRelationship> <InternationalAgent2> with
<Force>}

<InternationalAgentType> =

{terrorist group, dissident group, political party,
humanitarian organization}

Figure 3. A parameterized question suitable for generating sample questions and test questions.

In the following section we analyze how prior ontology – what was available before SQs, TQA and TQC were released – supported the development of the Teknowledge and SAIC systems. The former system was based on Cyc, and much of its development was done at Cycorp, so we call it Cyc/Tek here. The SAIC system was a collection of component systems, none of which answered all the questions in any test batch. The one we analyze here, developed by SRI International, answered roughly 40 of the 110 questions in each batch; we lack data for the other components of the SAIC system. To compare the Cyc/Tek and SRI systems properly we will report two sets of results for Cyc/Tek, one for all the test questions and another for the subset of questions answered by the SRI system.

The Cyc/Tek and SRI systems also differed in the prior ontologies available to them. Long before testing began, Cycorp, the developers of Cyc, released their *upper ontology* (UO), which contains very general class names; subclass relationships; instance-type relationships; relation names and their argument types; function names, their argument types, and the types of value they return; as well as English documentation of every class, function and relation; and a mapping to terms in the Sensus ontology developed by ISI.

Whereas the SRI team had access to the UO, only, Cyc/Tek had access to all of Cyc.

Results

The performance of the Teknowledge and SAIC integrated systems is analyzed thoroughly in (Cohen et al., 1998). Performance is not the focus of this paper – support provided by ontologies is – but two results are germane here: Both systems performed better on the sample questions (SQs) than on TQA, and both performed better when re-tested TQA and TQC than on the original tests performed four days earlier. In the four days between test and retest, significant improvements were made to the

systems. The question is, how much did the prior ontologies help in making these improvements?

We present results for two kinds of knowledge development. One is the development of knowledge sufficient to encode in a formal language the test questions in each batch, the other is the development of knowledge to answer the test questions. Results for the former are summarized in Table 1. The columns of the table represent the SRI system, which was tested on roughly 40 questions in each batch of 110; the Cyc/Tek system tested on the same questions as the SRI system; and the Cyc/Tek system tested on all 110 questions in each batch. Three numbers are reported for each system: *n* is the number of terms needed to encode all the questions attempted (i.e., roughly 40 or 110); *k* is the number of terms available in a prior ontology; and *s* is the ratio of *k* to *n*. The rows of Table 1 represent the batches of questions and the help provided by different prior ontologies. For example, the notation SQ | UO means “the help provided by the upper ontology (UO) in encoding the sample questions (SQ).” One can see in this row that SRI needed 106 terms to encode roughly 40 of the sample questions, and 22 of these terms were found in the UO, so the help provided by the UO is $22/106 = .21$. Encoding the questions in SQ required a number of terms to be added to the ontologies, and these terms were available to help encode questions in TQA and TQC. The notation TQA | UO denotes the help provided by the UO *only*, whereas TQA | SQ denotes the help provided by *everything encoded up through SQ*. Similarly, TQC | TQA denotes the help provided in encoding the questions in TQC by the terms in the ontology including those defined for SQ and TQA. Because the Cyc ontology is cumulative, these conditions – in which terms defined for earlier test questions are used to encode later test questions – are reported in rows labeled “Cyc” in Table 1. For instance, 418 terms were required by Cyc/Tek to encode the 110 questions in TQA, 402 of them were available in Cyc, including some defined when the sample questions SQ were added. Note that SRI did not have access to Cyc, so all rows in which questions were encoded with the help of Cyc are marked n/a for SRI.

| | SRI | | | Cyc/Tek(40) | | | Cyc/Tek(110) | | |
|-----------|-----|-----|-----|-------------|-----|-----|--------------|-----|-----|
| | n | k | s | n | k | s | n | k | s |
| SQ UO | 104 | 22 | .21 | 201 | 75 | .37 | 377 | 126 | .33 |
| SQ Cyc | n/a | n/a | n/a | 201 | 153 | .76 | 377 | 280 | .74 |
| TQA UO | 104 | 20 | .19 | 168 | 67 | .4 | 418 | 126 | .30 |
| TQA SQ | 104 | 81 | .78 | n/a | n/a | n/a | n/a | n/a | n/a |
| TQA Cyc | n/a | n/a | n/a | 168 | 168 | 1.0 | 418 | 402 | .96 |
| TQC UO | 106 | 16 | .15 | 277 | 81 | .29 | 402 | 131 | .33 |

| | | | | | | | | | |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| TQC TQA | 106 | 82 | .77 | n/a | n/a | n/a | n/a | n/a | n/a |
| TQC Cyc | n/a | n/a | n/a | 277 | 270 | .97 | 402 | 395 | .98 |

Table 1. Support (s) provided by ontologies for the development of problem solving systems to answer batches of test questions.

Cyc/Tek had higher support numbers in all conditions than SRI, meaning they found more terms in their prior ontologies than SRI did. However, we have broken the data into support provided to Cyc/Tek by *all* of Cyc vs. support provided by just the upper ontology, which SRI had. For example, the first row of Table 1 shows that to encode roughly 40 sample questions, SRI required 104 terms of which it found 22 in the UO; whereas Cyc/Tek required 201 terms to encode the *same* questions, and found 75 in the UO. Similarly, Cyc/Tek required 377 terms to encode all 110 sample questions, and found 126 in the UO.

Cyc/Tek required more terms to encode test questions (3.62 terms/question) than SRI (2.61 terms/question, and got more support from prior ontologies. For example, for Cyc/Tek to encode the roughly 40 questions in the TQA batch that SRI encoded, they required 168 terms, all of which existed in the Cyc ontology.

In one respect, the SRI and Cyc/Tek results are very similar. The reuse rate of terms *not* in the upper ontology – terms in Cyc or terms developed for earlier batches of test questions – was 60%-65% for both SRI and Cyc/Tek, across question batches TQA and TQC. This result is shown in Table 2. The columns in this table represent the number of terms needed to encode a test batch, N; the number found in the upper ontology, K(UO); the number found elsewhere, K(other); and the ratios of K(UO) and K(other) to N. That is, the support provided by terms in the upper ontology is $s(UO)=K(UO)/N$, while the support provided by other prior ontology is $s(other)=K(other)/N$. Note that $s(other)$ ranges from .59 to .68 for test batches TQA and TQC. In fact, the overall reuse of non-UO terms for Cyc/Tek and SRI was .66 and .60, respectively; whereas the overall reuse of UO terms for Cyc/Tek and SRI was .32 and .17, respectively. Thus, much of the difference in reuse statistics between SRI and Cyc/Tek is due to their exploitation of the upper ontology. Said differently, 22% of the terms SRI reused came from the upper ontology while the figure was 33% for Cyc/Tek.

| | N | K(UO) | K(other) | S(UO) | S(other) |
|------------------|-----|-------|----------|-------|----------|
| SRI TQA | 104 | 20 | 61 | .19 | .59 |
| SRI TQC | 106 | 16 | 66 | .15 | .62 |
| Cyc/Tek TQA(40) | 168 | 67 | 101 | .40 | .6 |
| Cyc/Tek TQC(40) | 277 | 81 | 189 | .29 | .68 |
| Cyc/Tek TQA(110) | 418 | 126 | 276 | .30 | .66 |
| Cyc/Tek TQA(110) | 402 | 131 | 264 | .33 | .66 |

Table 2. Support provided by terms in UO and terms from other prior knowledge bases and ontologies.

In addition to encoding test questions, Cyc/Tek and SRI developed knowledge to answer the questions. This knowledge, called *axioms* generically, is composed of terms, so we can ask how prior ontologies helped the development of axioms. As before the relevant metric is $s(i,j)=k(i,j)/n(i)$, only here, $n(i)$ denotes the number of terms required to encode the i th axiom.

SRI provided data on how ontologies supported writing axioms. The rows of Table 3 represent the phases of the experiment and the source of prior ontology. The first row, SQ | UO shows that 1703 axioms were encoded to solve the sample questions SQ, and these axioms required 461 terms, of which 51 were in the upper ontology, UO, for a support value of 0.11. The second row shows that in the four days between the test and retest on batch TQA, 123 axioms were encoded, requiring 195 terms. 30 of these terms were found in the UO. The third row shows that 109 of the 195 terms were found in *all* the ontology developed prior to the test on TQA, namely UO and SQ. A comparison of the second and third rows shows that $109-30=79$ reused terms came from SQ. The same pattern repeats in the two remaining phases of the experiment: After the scenario modification but before TQC, 1485 axioms were added to the SRI system. These required 583 terms of which 40 existed in the UO and 254 were found in the UO, SQ, and TQA prior ontologies. Similarly, between the test and retest on TQC, 215 terms were required for 304 axioms; only 24 of these existed in the UO, and 100 more were found in the ontologies developed after the UO.

It is unclear why prior ontologies provided significantly less support for encoding axioms than for encoding test questions. In both cases the support came in the form of terms, but why are the terms required to define axioms less

likely to be in a prior ontology than the terms needed for test questions? One possibility is that test questions include fewer terms that represent *individuals* (e.g., #HassiMessaoud-Refinery) than do axioms, so terms in test questions are less specific and more likely to exist in a prior ontology than terms in axioms. We will be looking at our data more closely to see whether this is the case.

| | SRI | | | |
|------------------------------|--------|-----|-----|-----|
| | Axioms | n | k | s |
| SQ UO | 1703 | 461 | 51 | .11 |
| From TQA to TQA retest UO | 123 | 195 | 30 | .15 |
| From TQA to TQA retest SQ | 123 | 195 | 109 | .56 |
| From TQA retest to TQC UO | 1485 | 583 | 40 | .09 |
| From TQA retest to TQC TQA | 1485 | 583 | 254 | .44 |
| From TQC to TQC retest UO | 304 | 215 | 24 | .11 |
| From TQC to TQC retest TQC | 304 | 215 | 124 | .58 |

Table 3: SRI measured the number of terms required to add problem-solving axioms to their system, and the reuse of terms from the UO and subsequent ontology efforts.

Discussion

Does prior knowledge in ontologies and domain-specific knowledge bases facilitate the development of knowledge-based systems? Our results suggest that the answer depends on the kind of prior knowledge, who is using it, and what it is used for. The HPKB upper ontology, 3000 very general concepts, was less useful than other ontologies, including Cyc and ontologies developed specifically for the crisis management domain. This said, Cyc/Tek made more effective use of the upper ontology: 33% of the terms it reused came from there whereas 22% of the terms SRI reused came from the upper ontology. Why is this? One reason is probably that Cycorp developed the upper ontology and was more familiar with it than SRI. Knowledge engineers tend to define terms for themselves if they cannot quickly find the terms in an available ontology. Once this happens – once a term is defined anew instead of reused – the knowledge base starts to diverge from the available ontology, because the new definition will rarely be identical with the prior one. Another reason for disparity in reuse of the upper ontology is that SRI preferred their own definitions of concepts to the available ones. We lack the data to assess which of these explanations accounts for most of the disparity.

As to the uses of prior knowledge, our data hint at the possibility that prior knowledge is less useful for encoding axioms than it is for encoding test questions.

Whereas reuse of the upper ontology depends on who is using it, other ontologies seem to account for a roughly constant (60% – 66%) rate of reuse, irrespective of who developed these ontologies. For SRI, these ontologies were just those developed for batches of questions SQ, TQA, TQB, TQC and TQD. To be concrete, the 60% of the terms required for TQC were defined while encoding SQ, TQA and TQB. The picture is a bit cloudier for Cyc/Tek because they had the Cyc ontology throughout, and we lack the data to say whether the 66% non-UO reuse came from terms defined for previous batches or from Cyc.

Despite this ambiguity we speculate that in the process of building a domain-specific knowledge-based system, the rate of reuse of terms defined earlier in the process is 60%-70%. Whether this figure is due to a few terms being reused very frequently or many terms being reused moderately, we do not have the data to judge. Although the rate of reuse of terms from very general ontologies may be significantly lower (e.g., 15%–30%), the real advantage of these ontologies probably comes from helping knowledge engineers organize their knowledge bases along sound ontological lines. However, we can offer no data pertinent to this use of general ontologies.

Conclusion

Many questions remain. Our data are crude summaries of reuse of terms, they do not tell us much about the work that knowledge engineers do when they build domain-specific knowledge bases. How long will a knowledge engineer hunt for a relevant term or axiom in a prior ontology? How rapidly do knowledge bases diverge from available ontologies if knowledge engineers don't find the terms they need in the ontologies? By what process does a knowledge engineer reuse not an individual term but a larger fragment of an ontology, including axioms? How does a very general ontology inform the design of knowledge bases, and what factors affect whether knowledge engineers take advantage of the ontology? Why do prior ontologies apparently provide less support for encoding axioms than for encoding test questions? Finally, will the results we report here generalize to domains other than crisis management and research groups other than SRI and Cyc/Tek? We expect to answer some of these questions retrospectively by analyzing other data from the first year of the HPKB program and prospectively by designing experiments for the second year.

References

Paul Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Gunning, and Murray Burke. The DARPA High Performance Knowledge Bases Project. AI Magazine, Winter, 1998. pp. 25-49

From Visual to Logical Representation: A GIS-Based Sketching Tool for Reasoning about Plans

John Li, Cleo Condoravdi, Adam Pease

[jli, ccondora, apease@teknowledge.com

Teknowledge Inc.

Abstract

Abstract: Multi-modal and heterogeneous logic reasoning is of increasing importance within the AI community. The GIS based ArcView COA Sketcher (ArCS) sketch and translation tool developed under DARPA's High Performance Knowledge Bases program is an example of an enabling tool towards that goal. Army Course of Action (COA) sketches can be drawn and translated automatically into statements in a formal logic with the tool. The statements are inputs to a geographic reasoner as well as systems reasoning about plans. This paper discusses the design of the sketching tool, and issues in creating an effective correspondence between the visual and logical representations of a COA.

Introduction

Multi-modal and heterogeneous logic reasoning is of increasing importance within the AI community. Visual presentation of information through diagrams, sketches and charts is so ubiquitous in human communication that it has long been desired to have automated reasoning systems taking visual representations as inputs. More generally, research effort has been devoted to developing logics for diagrammatic reasoning. For example, [Fisler, 1996] has developed a heterogeneous logic for hardware verification of design diagrams. A visual representation has also been used in the teaching of classical logic itself [Barwise & Etchemendy, 1995].

DARPA's High Performance Knowledge Bases (HPKB) program provided us with an excellent opportunity for developing tools for visual inputs to systems performing knowledge-rich symbolic reasoning. The purpose of the HPKB program is to advance the state of the art in knowledge representation and reasoning and create applications incorporating advanced AI techniques that are relevant to the military [Cohen et al 1998]. One application of this program is to develop systems for evaluating and critiquing operational army plans for courses of action on the battlefield.

Army Courses of Action (COAs) are high-level battle plans. They describe the intended actions of friendly troops on the battlefield on the basis of possible enemy deployment and actions. Human planners specify COAs by means of a sketch and a textual description in accordance with army practice [Army, 1997].

The sketch consists of standardized symbology placed on a map. It presents map-based information about

the battlefield — the location of friendly and enemy troops, terrain features and obstacles, military regions and boundaries, possible battle positions and maneuver paths — as well as the planned tactical actions.

The textual description describes in a controlled English grammar the actions of the troops, their temporal relation and their intended purpose within the overall plan.

The COA statement and the COA sketch are in part complementary to each other but there is also significant overlap between them. As a result, the two parts of the COA need to be combined through a fusion process.

COA Sketches as Inputs to Plan Critiquers

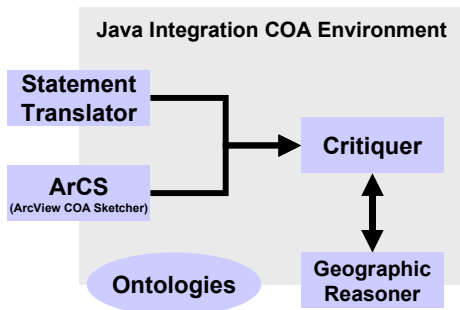
The process of getting from the original COA inputs (statement and sketch) to the formal inputs needed for the critiquing systems presents challenges that are worth some discussion. One is that reasoning for plan evaluation relies on qualitative concepts and relative spatial relations between objects rather than the absolute geographical position of objects or the geographical profile of a region. However, it is the latter that is explicitly recorded in the sketch. For example, a critique checking to determine whether an action can successfully accomplish its stated purpose to enable another action will check whether the targets of the enabling action include all enemy troops on the path of the enabled action. This means that the reasoning system needs information making reference to concepts like spatial subsumption, betweenness along a non-cyclic path, trafficability of terrain, etc.

This implies that a translator should pass information about the absolute location of objects on to a geographic reasoner that can calculate qualitative spatial relations and estimate the trafficability of regions. Moreover, many concepts that are important to critiquers, such as the forward edge or the rear area of the battlefield, although implicitly depicted in the sketch, do not correspond to any specific symbol and need both geographic reasoning and domain specific knowledge to be identified with the proper region on the map.

Our GIS based ArcView COA Sketcher (ArCS) sketch and translation tool was, therefore, designed to provide formal inputs to a geographic reasoner as well as systems reasoning about plans. COA sketches can be drawn and translated automatically into statements in a formal logic with the tool. In what follows, we will first

discuss the software environment for COA critique and ArCS development. Then we will introduce an exemplar COA sketch drawn with ArCS and will use it in our discussion of the design of the translation process. Issues in creating an effective correspondence between the visual and logical representations of a COA are presented afterwards.

Execution Environment



We utilized Cyc [Lenat, 1995] as our knowledge representation and inference environment for the COA critique development. Relevant army doctrine is coded as logical statements in MELD/CycL [Cycorp, 1998]. Instances of COA statements and sketches are translated into MELD and imported into Cyc. Cyc is an inference system that can directly use these inputs to answer queries posed by the critiquer. Each COA critique was implemented as a *problem solving method* (PSM) [Gil, 1996] (this could be also termed a knowledge based procedure) and coded in Java. Each PSM made several calls to Cyc with a knowledge base query. Subsequent queries were created based on the results from earlier queries posed by the PSM. The most widely used PSM for COA critiquing could be called *critique and refine*. An initial query would be posed to determine if there was a problem of a particular sort. Further queries would zero in on the specifics of the problem so that the answer returned could provide the user with enough information necessary

to effect a repair.

We utilized GeoRep [Ferguson, et al, 1999] as our geo-spatial reasoner. GeoRep can take as inputs the latitudes and longitudes of all objects drawn on the COA sketch and answer location and proximity questions with its qualitative spatial reasoning. It also provides trafficability support. For details see [Donlon et al, 1999].

We used the ArcView Geographic Information System [ESRI, 1996:1] as our sketching environment. The Avenue [ESRI, 1996:2] scripting language was used both for handling user interaction and creating the MELD logical statement output created from the sketch. The sketch outputs in MELD followed the common ontology developed for the domain and were used by GeoRep, the Cyc-based critiquer and the critiquing systems of GMU (built with Disciple KA shell [Tecuci et al, 1999]) and ISI (built with the expect KA shell [Swartout & Gil, 1996]), and UMass's knowledge-based war gaming simulator [Atkin et al, 1999].

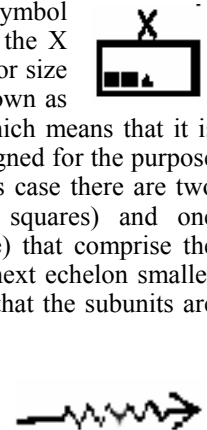
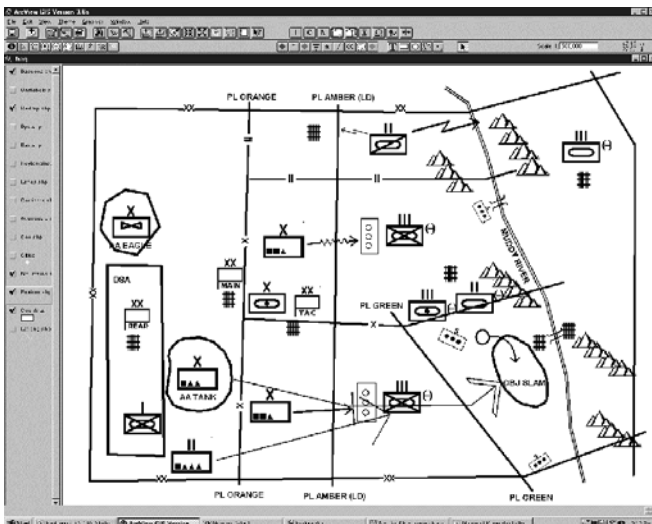
COA Example

The graphic above shows a standard army COA drawn within ArCS. A human planner uses the ArCS tool to depict the deployment of friendly and enemy forces, the actions undertaken by friendly forces, the organization of the battlefield, including the areas of responsibility for friendly forces, and various features of the terrain. When the planner finishes the sketch, a click of the translation button allows thousands of logical statements to be generated instantly. This specific COA sketch, for instance, was translated into about 4,000 MELD statements.

To fulfill the drawing and translation function of ArCS, we made TrueType fonts for composable military symbols, and defined a mapping between the military symbology and ArcView's drawing functions, as well as another mapping between the military symbology and the MELD logical statements.

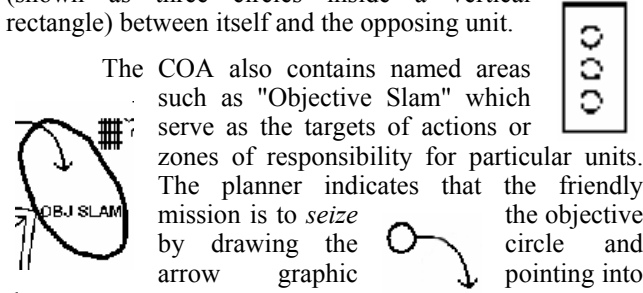
For example, the box symbol signifies a friendly military unit with the X denoting that the unit has the *echelon* or size of *brigade*. This particular unit is known as a *task organized composition unit*, which means that it is composed of subunits temporarily assigned for the purpose of a battle or larger operation. In this case there are two mechanized units (denoted by the squares) and one armored unit (denoted by the triangle) that comprise the brigade. The system knows that the next echelon smaller than brigade is *regiment* so it asserts that the subunits are regiment sized.

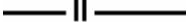
The arrows in the diagram denote actions that the units are



supposed to undertake. The symbol denotes that the unit at the tail of the arrow is performing a *fix* task against the unit at the head of the arrow. A fix task is one in which the object is to prevent the enemy unit from moving. This can be done by artillery fire. In this particular case, the enemy is further constrained by the presence of a minefield (shown as three circles inside a vertical rectangle) between itself and the opposing unit.

The COA also contains named areas such as "Objective Slam" which serve as the targets of actions or zones of responsibility for particular units. The planner indicates that the friendly mission is to *seize* the objective by drawing the circle and arrow graphic pointing into the area.



Another important class of graphics is the lines that divide the battlefield. For example, the line, which has two vertical bars on it in the top center of the sketch, indicates that the area of responsibility for the  battalion directly above it ends at the line. The vertical lines are called *phase lines* and provide an indication of the phasing of the plan. Phase lines indicate the position of units during different phases of the plan. The phase line acquires its full meaning when combined with temporal information provided in the COA statement. The vertical line which is marked "PL Amber" at the bottom center of the COA indicates that the line is named as indicated and is further specified as an "LD" or Line of Departure which denotes the point from which an attack takes place.



Translation of the Symbols

We first define a context or, in Cyc terminology, a *microtheory* in which all statements about the COA would be placed. This allows knowledge about the particular plan to be clearly differentiated from knowledge about the domain of all COAs. Also, by virtue of inserting the COA microtheory within the lattice of other Cyc microtheories, it allows for reasoning to be made more efficient by excluding inheritance from reasoning context which are irrelevant to the COA domain. In the MELD syntax below **Constant** introduces a new symbol to the system. **in Mt** indicates that all assertions below it are to be made within the given context. **F** indicates that a *formula* or logical statement is to follow. **isa** states that an instance is a member of a class. **genlMt** states that one microtheory inherits the contents of another.

```
Constant: BlueDivisionCOA1-1Mt.
in Mt: BaseKB.
F: (isa BlueDivisionCOA1-1Mt
```

```
COASpecificationMicrotheory).
F: (genlMt BlueDivisionCOA1-1Mt
ModernMilitaryTacticsMt).
```

The specification of the military units involved in the COA relies on the ontology of military units developed by [Andersen & Petersen, 1997]. Each unit has a specific military specialty, echelon, and strength. The relation *InstanceExistsCount* statements deserve further explanation. They are based on relations that are the MELD equivalents of frame predicates, which indicate that the relation given as the first argument holds between the instance given as the second argument and N instances of the class specified in the third argument where N is the fourth argument. This statement is equivalent to (following

notation in [Sowa, 1999])
 $(\exists x:\arg3)(x@arg4 \wedge arg1(arg2,x)).$

```
Constant: BlueMechBgd1.
in Mt: BlueDivisionCOA1-1Mt.
F: (isa BlueMechBgd1
MechanizedInfantryUnit-
MilitarySpecialty).
F: (isa BlueMechBgd1
ArmoredUnit-MilitarySpecialty).
F: (echelonOfUnit BlueMechBgd1
Brigade-UnitDesignation).
F: (sovereignAllegianceOfOrg BlueMechBgd1
Blue-Side).
F: (troopStrengthOfUnit BlueMechBgd1
RegularStatus).
F: (relationInstanceExistsCount
subOrgs-Direct
BlueMechBgd1
ArmoredUnit-MilitarySpecialty
1).
F: (relationInstanceExistsCount
subOrgs-Direct
BlueMechBgd1
MechanizedUnit-MilitarySpecialty
2).
```

The system also names the areas that the units are placed in and have responsibility over. In the statements following, the shape and geographic location of areas of interest are given. While the Cyc-based reasoner does not use these metric points directly, they are generated and passed to the geographic reasoner which turns the metric statements into propositional, or relative statements about the geographic entities and passes those statements back to the critiquer.

```
Constant: Loc36.
in Mt: BlueDivisionCOA1-1Mt.
F: (isa Loc36 GeographicalRegion).
F: (objectFoundInLocation BlueMechBgd1 Loc36).
F: (shape Loc36 (AbstractFn Square)).

F: (longitude (CenterFn Loc36)
(Degree-UnitOfAngularMeasure -97.5515)).
F: (latitude (CenterFn Loc36)
(Degree-UnitOfAngularMeasure 37.8412)).
```

```

Constant: P241.
F: (isa P241 GeographicalThing).
F: (longitude P241
  (Degree-UnitOfAngularMeasure -97.5087)).
F: (latitude P241
  (Degree-UnitOfAngularMeasure 37.7984)).

;; etc (similar for other points)

F: (pointsOfBorder (BorderFn Loc36)
  (TheList P241 P242 P243 P244)).

```

Note that a lengthy specification of the coordinates specifying the center and the termini of the region (P241, P242, P243, and P244) is necessary for the geographic reasoner to recognize the shape of the region. These specifications for the regions, as well as those for the polylines and other geometric shapes, are the inputs to the geographic reasoner only. The critiquing systems use the derived results from the geographic reasoner such as the statements about the shape and degree of trafficability of a region given in the code below.

Because ArCS is built on top of a geographic information system, it is possible to overlay the sketch on digital terrain data and to use that information for further processing. One of the results of such processing [Donlon et al, 1999] is to generate information about the trafficability of regions. We can then attach to polygons that define those regions information about their trafficability.

```

Constant: COO2.
in Mt: BlueDivisionCOA1-1Mt.
F: (isa COO2 GeographicalRegion).
F: (degreeOfTrafficability COO2
  TerrainSeverelyRestricted).
F: (shape COO2 (AbstractFn Polygon)).

```

The sketch tool also provides the human planner with the ability to describe features that do not exist at the beginning of the plan but rather come into being as the plan unfolds, such as candidate battle positions. For example,

```

Constant: BP10.
in Mt: BlueDivisionCOA1-1Mt.
F: (isa BP10 GeographicalRegion).
F: (candidateBattlePositionOfCOA
  BlueDivisionCOA1-1Mt BP10).
F: (shape BP10 (AbstractFn Polygon)).

```

As discussed above, the phase lines and lines of responsibility collectively define regions that units are responsible for controlling at different points in time. Temporal information is not generated from the sketch and as a result not shown here. However, the process of translating the statements and combining them with the

ArCS output does result in temporal information being provided.

```

Constant: Loc54.
in Mt: BlueDivisionCOA1-1Mt.
F: (isa Loc54 PhaseLineBoundedArea).
F: (sectorOfResponsibility BlueMechBgd1
  Loc54).
F: (hasPartAsBorder Loc54 BoundaryBtnS).

```

The tasks assigned to units are also produced from the sketch and further specified in the text.

```

Constant: Fix1.
in Mt: BlueDivisionCOA1-1Mt.
F: (isa Fix1 Fix-MilitaryTask).
F: (unitAssignedToTask Fix1 BlueMechBgd1).
F: (objectActedOn Fix1 RedMechRegt1).

```

Issues in Translation

The representations provided above are relatively simple. The challenge was to define representations that could be easily and unambiguously generated by the sketch tool and yet still be powerful enough to represent all the information of interest and be merged successfully with the more complicated representations generated from parsing the controlled English COA text.

An additional complexity existed with regards to the domain. Because military COAs have been intended for understanding and use by humans, they contain considerable ambiguity and flexibility. For example, the presence of a unit at a particular point on a COA sketch does not necessarily mean that the unit will in fact be located at that precise point at any time during the battle. It is merely a candidate position that is consistent with the intent and responsibilities of the planner. The challenge was to provide the utility of such intended flexibility while not complicating the system design. In the end, this was handled by ensuring that the reasoning that operated on the sketch outputs was not sensitive to the exact positions. In this sense, the critiquing rules embodied a tolerance for imprecision. Accomplishing this required incorporating geographic reasoning which asserted qualitative statements about spatial relations as described earlier.

A similar consideration was that the units themselves are intended as prototypes or descriptions of units rather than references to real world units. For example, by placing a mechanized division symbol on the map, the human planner is not asserting that a particular division such as the 5th Mechanized division will be positioned at that location but rather that any available mechanized division can play that role in the plan. There again, the critiquing rules needed to take this into account.

Related Work

The nuSketch tool of Northwestern University [Forbus et al, 2000], under development during the HPKB program, has been intended to have a similar functionality to ArCS. It uses in addition a combination of speech and gesture to specify COA elements in order to free commanders from mouse and menus. Important differences include the fact that the ArCS system used a commercial geographic information system as its base.

Conclusion

The work described here was part of the effort to provide formal inputs to the various reasoning systems. We created a rich sketching tool that allowed a user to specify a battlefield plan in standard Army symbology. That symbology was converted to a logical representation, which combined with statement translation, could be effectively employed for reasoning about the suitability, feasibility and correctness of a military Course of Action.

Acknowledgements

We would like to acknowledge our DARPA sponsor, Murray Burke for supporting this effort. We would also like to acknowledge inspiration for, collaboration with and testing of ArCS from Ken Forbus and his team at Northwestern University. We would like to acknowledge Henry Gunthardt, SAIC, for his contribution in coding a drawing function.

References

Andersen, W. & Petersen, B. 1997, "Military Units Ontology", unpublished report.

Army field manual, FM 101-5. Headquarters, Department of the Army, Washington, DC, 31 May 1997.

Atkin, Marc, David L. Westbrook and Paul R. Cohen. 1999. Capture the Flag: Military Simulation Meets Computer Games. AAAI-99 Spring Symposium on AI and Computer Games.

Barwise, J. and Etchemendy, J. 1995, Hyperproof. Stanford CSLI Publications. Cambridge University Press.

Cohen, P., Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Gunning, and Murray Burke 1998. The DARPA High Performance Knowledge Bases Project. AI Magazine, Winter, 1998. pp. 25-49.

Cycorp, 1998, "Features of the CycL Language", on-line report at <http://www.cyc.com/cycl.html>.

Donlon, J.J. and Forbus, K.D. 1999, Using a Geographic Information System for Qualitative Spatial Reasoning about Trafficability. Proceedings of the Qualitative Reasoning Workshop. Loch Awe, Scotland.

ESRI, 1996:1, "ArcView GIS: Using ArcView" ESRI software documentation, Redlands, CA.

ESRI, 1996:2, "Avenue: Using Avenue", ESRI software documentation, Redlands, CA.

Ferguson, R.W. and Forbus, K.D. 1999, GeoRep: A Flexible Tool for Spatial Representation of Line Drawings. Proceedings of the Qualitative Reasoning Workshop. Loch Awe, Scotland.

Fisler, K. 1996, A Unified Approach to Hardware Verification Through a Heterogeneous Logic of Design Diagrams. PhD Dissertation. Indiana University Department of Computer Science, August.

Forbus, K., Ferguson, R., Usher, J., 2000, "Toward a Computational Model of Sketching", in preparation for AAAI-2000.

Gil, Y., and Melz, E., 1996, "Explicit Representations of Problem-Solving Strategies to Support Knowledge Acquisition". Proceedings of the Thirteen National Conference on Artificial Intelligence (AAAI-96), Portland, OR, August 4-8, 1996.

Lenat, D., 1995, "Cyc: A Large-Scale Investment in Knowledge Infrastructure." Communications of the ACM 38, no. 11, November.

Sowa, 1999, "Knowledge Representation: Logical, Philosophical and Computational Foundations", Brooks/Cole pub.

Swartout, W. and Gil, Y., 1996. "EXPECT: A User-Centered Environment for the Development and Adaptation of Knowledge-Based Planning Aids". In *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, ed. Austin Tate. Menlo Park, Calif.: AAAI Press.

Tecuci G., Boicu M., Wright K., Lee S.W., Marcu D., and Bowman M., 1999, "An Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, July 18-22, Orlando, Florida, AAAI Press, Menlo Park, CA.

Using Large Ontologies to Enable Semantic Interoperability of Problem Solvers

Adam Pease
Teknowledge
apease@teknowledge.com

Abstract

A key to delivering a scalable solution for the integration of knowledge based problem solvers is the notion of semantic integration. We show how semantic integration using a very large knowledge base provides particular leverage in achieving a robust solution. We describe some existing approaches to this problem and then detail our solution applied to several knowledge based problem solvers. We conclude with lessons learned and directions for future research.

Introduction

In this paper, we will explain the notion of semantic integration. We will show this in the context of our work on the DARPA High Performance Knowledge Bases (HPKB) program. The goals of HPKB are to advance the state of the art in knowledge based problem solving, knowledge representation, reasoning and knowledge capture.

The HPKB program participants are constituted into three groups: Challenge Problem Developers, Integrators and Technology Developers. The Challenge Problem Developers are charged with creating test problems which address tasks which are relevant to DARPA's military customers and are amendable to a knowledge based solution. The Challenge Problems (CPs) must balance several objectives including creating a problem which is solvable and yet challenging enough such that it forces the application and development of new technologies. Integrators are charged with solving the CPs and applying whatever technologies are needed to accomplish that task. Technology developers are charged with supplying the integrators with the component technology solutions.

In our role as an integrator we are tasked in part with creating an integrated architecture which allows domain specific problem solvers to work together and contribute to an overall knowledge based solution. Our basis for this integration is the use of a single very large knowledge base. In this first year of the program we have successfully integrated several problem solvers using the Cyc system (Lenat, 1995; Lenat & Guha, 1990).

Background

Software integration is a difficult problem. Systems that were not designed to work together always have a host of different assumptions.

We will divide integration into three layers of connectivity issues. The first is what we will call transport layer issues. This concerns the mechanisms of how to get the bits from one process or machine to another. Solutions include sockets, RMI and CORBA. Another set of issues are found at what we'll call the syntax layer. This concerns how to convert number formats, "syntactic sugar" or the labels of data. The more challenging task is to deal with what we will call semantic connectivity. The integrator must understand the meaning of each data element. Considerable related work has occurred in the database community on the issue of integrating databases at the semantic level [Wiederhold, 1996].

The current state of the practice in software integration consists largely of interfacing pairs of systems as needed. We term this pairwise integration. It is a problem because pairwise integration doesn't scale up. Unanticipated uses are hard to cover later. Chains of integrated systems evolve at best into stovepipe systems. Each integration is only as general as it needs to be to solve the problem at hand.

Some success has been achieved in low level integration and reuse. Systems which use the same scientific subroutine libraries or graphics packages at least are forced into similar representational choices for low level data. DARPA has also invested in early efforts to create large reuse libraries which can assist in integrating large systems at higher levels (Carrico, 1997). Considerable work has gone into expressing a generic semantics of plans in an object oriented format (Pease & Carrico, 1997; Pease & Carrico, 1997:2). Additional work in applying that generic semantics to domain specific applications is promising (Pease & Albericci, 1998).

The development of ontologies for integrating manufacturing planning applications (Tate, 1998) and workflow (Lee, 1996) have also been ongoing.

Another option for semantic integration is to perform software mediation (Park et al, 1997). This could be seen as a variant on pairwise integration, but because integration is done by knowledge based means, there is an expression

of the explicit semantics of the conversion. This renders the effort more reusable.

Personnel at Kestrel Institute have been successful at defining formal specifications for data and using those theories to integrate formally specified software [Srinivas & Jullig, 1995]. In addition, personnel at Cycorp have successfully applied Cyc to the integration of multiple databases.

The Solution

We used a large general purpose knowledge base (Cyc) to link problem solver input and output to the semantic concepts the input and output denote. We augmented the knowledge base with domain or problem specific knowledge to support each specific problem solver. We used knowledge base axioms to create a declarative specification of how problem solver concepts map onto the knowledge base. This approach will now be described for several problem solvers.

Network Flow Problem Solver

One of the HPKB challenge problems can be broadly characterized as providing a battlefield commander with a knowledge level analysis of the characteristics of the battlefield. One element of this is determining how and to what extent goods and personnel can be transported from one location to another. To solve this problem we integrated a Network Flow Problem Solver (NFPS) supplied by personnel at Kestrel Institute.

In the first application of the problem solver we needed to supply a set of roads and the cities they connect and the carrying capacities of the roads in number of vehicles per hour. Note that in this example we were not concerned with sophisticated traffic flow estimation which includes local factors such as temporary backups and congestion. After processing, we wanted the output to be a list of the actual capacities of each road given that each exists as part of a total system. Figure 1 shows an example with capacities on each link. A is given as an infinite source and E is an infinite sink.

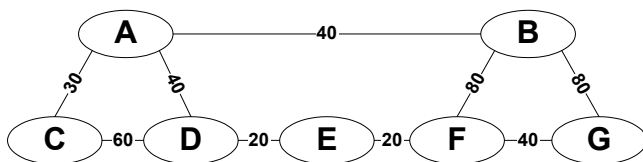


Figure 1 - Network Capacities

Figure 2 shows the same network with the situated capacities on each link. This graph reflects the fact that the only links to node E are bottlenecks. Therefore, even

though link B-F can carry 80 units per time period, the

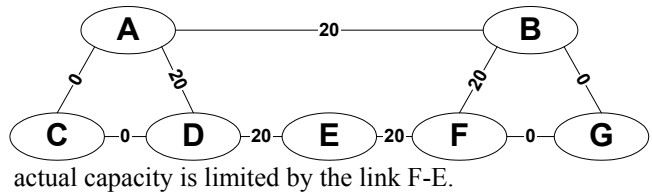


Figure 2 - Situated Capacities

Note that the NFPS has no knowledge of roads or vehicles, only abstract notions of nodes, arcs, capacities and situated capacities. Our next step was to link these abstract notions with concrete notions of roads and vehicles.

Cycorp was already in the process of creating a detailed abstract theory of networks and had already created concepts for roads, vehicles and the flow of traffic. A crucial concept which was not present at first was the notion of a situated capacity as described above..

Our next step in integration was to create a system executive and GUI which allowed user to enter and visualize road information, transformed the road information into input for the NFPS, retrieved the output from the NFPS and then made assertions to the Cyc KB based on that output.

Figure 3 shows the output of NFPS after processing by the system executive. The statements are sent to Cyc. Ellipses denote repetition of the same type of statement for all the nodes or arcs in the network. #isa is the Cyc instance-of relationship in which an individual is declared to be a

```
(#isa      #Network1
  #NetworkFlowSystem-Bounded)
(#$flowTypeOfSystem    #Network1
  #CarPerHour)
(#isa #Arc1 #Path-Simple)
...
(#$linkFromToInSystem    #Arc1
  #Node1
  #Node2
  #Network1)
...
(#$equals  (#$LocalLinkFlowCapacityFn
  #Arc1 #Network1)
  40))
...
(#$equals  (#$SituatedLinkFlowCapacityFn
  #Arc1 #Network1)
  20)
...
```

member of a class.

Figure 3 - NFPS output to Cyc

A block diagram of the entire system is shown in Figure 4.

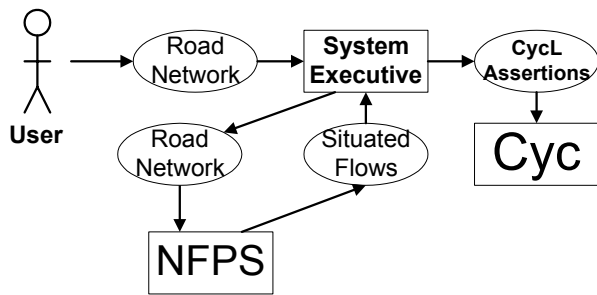


Figure 4 - Network Flow Problem Solver block diagram

Dynamical Recognizers

The central thesis of this technology development task is that representations of dynamics are the foundation for knowledge level coding of verbs (Cohen, 1998). While the first iterations of the challenge problem do not truly stress this technology, an initial integration provided practical feedback on its eventual application as part of a sophisticated knowledge based system.

The battlespace challenge problem provides a large data set describing the movements of battlefield entities over a several hour period. The first level of understanding the significance of the data is to classify the vehicles and battlefield sites based the movement data. We integrated several recognizers that use statistical pattern matching to identify patterns in the data. Each recognizer identifies one

```

(#$latitude      #$Refueling34
  ($Degrees 54))
(#$longitude     #$Refueling34
  ($Degrees 25))
(#$after
  ($StartFn      #$Refueling34
    ($DayFn 14
      ($MonthFn  #$February
        ($YearFn 1996))))))

```

type of physical battlefield location. Two of these locations are refueling stations and command posts.

The statements in Figure 5 assert the location of a particular site and the time at which its existence began.

Figure 5 - Sample Site Recognizer Assertions

Workarounds Inputs

The workarounds challenge problem is another part of the overall battlespace problem for HPKB. It addresses the task of finding engineering workarounds for a disrupted transportation network. When confronted with a damaged bridge or road, military engineers must find repair solutions which address the capabilities of the vehicles which need to pass, the available engineering assets, and the characteristics of the terrain and damaged transportation infrastructure.

This problem was posed by the Challenge Problem Developers as a database which characterizes the damaged infrastructure, engineering assets and terrain. Our integration task was to transform that database into a set of assertions to a knowledge base and to link that to knowledge based workaround reasoner which is built in Cyc. That reasoner then performs workarounds problem solving to recommend a solution.

The database has several records which are utilized in specifying the current problem (from [Jones, 1998]):

1. Unit. A military unit with its nominal and actual properties
2. Vehicle-Of. Assigns vehicle counts to Units (by type of vehicle)
3. Equipment-Of. Assigns equipment counts to Units (by equipment type & vehicle type)
4. Unit-Of. Assigns specific Units as parts of other Unit
5. Site. A geographically located militarily-relevant place
6. Site-Point. A point that is part of the definition of site geometry
7. Site-Part. Assigns specific Features/Sub-Sites to some Site
8. Site-Attrib. Free-form attributes for Sites
9. Damage-Specific degradation of some Site/Feature
10. Damage-Attrib. Free-form attributes for Feature-Damage

The workarounds problem solver first maps these database records into concepts in the knowledge base. It then poses hypothetical actions to Cyc for a determination of whether the actors in the problem are capable of the actions. Actions may need to be decomposed or enabled by the results of other actions. A block diagram of the system is given in Figure 6.

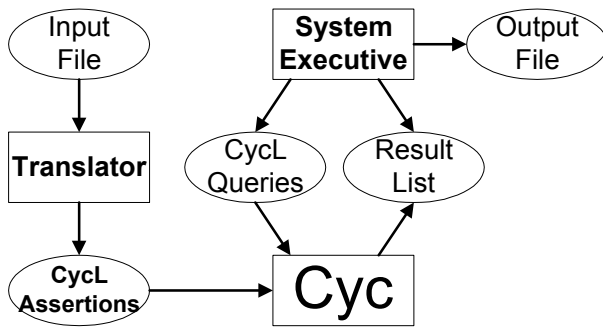


Figure 6 - Workarounds Problem Solver

Future Work

The real payoff in semantic integration will come in later phases of this project when each of the systems described above are linked into a single tool suite for battlespace problem solving. Before this paper is in print we will have received several more products to be integrated. Among them are tools for terrain analysis, trafficability analysis and route finding. We will also be integrating a commercial Geographic Information System (GIS).

The terrain analysis tool will examine terrain information supplied by the GIS and produce a set of polygonal regions which are labeled as to their characteristics relative to movement actions. This analysis is independent of vehicle type or other factors in the situation.

The trafficability analysis tool will relate the results of the terrain analysis tool to the specific situation given in the challenge problem. It will provide estimates of movement costs both in time and resources used for specific vehicles, combat and weather conditions.

The route finding tool will work with the outputs from the trafficability tool. It looks at the list of movement costs as an abstract graph and finds a set of the best paths through the network with regards to those costs. Those paths can then be added to the transportation network which is the input to the Network Flow Problem Solver described previously.

While the flow of information for these tools is relatively sequential, the outputs of each are relevant for the workarounds problem solver. By virtue of having integrated each tool with the Cyc KB, there will be no need to have a separate integration with the workarounds problem solver. The integration will already have been done.

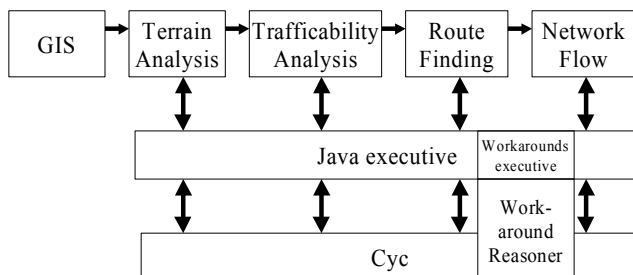


Figure 7 - Full Battlespace System

Conclusion

In this paper we have attempted to show the process of achieving semantic integration by describing the integration of several problem solvers. We have shown how these problem solvers are specialized for a particular problem. We have alluded to the benefits provided by our integration approach over a more conventional pairwise integration.

Acknowledgements

Thanks first to Doug Lenat for his years of work in leading the development of Cyc, without which, our integration effort would not be possible. Thanks to Fritz Lehmann and Ming Xu at Cycorp for creating the network axioms used with NFPS, to Cleo Condoravdi and John Li at Teknowledge for their excellent work on integration, Paul Cohen and his group at UMass for their help in integrating the dynamical recognizers, and Lee Blaine and John Anton at Kestrel for their help in integrating NFPS. Thanks also to Dave Gunning, the DARPA HPKB program manager for creating and supporting the program which made this work possible.

References

- Carrico, T., (1997) Object Model Working Group, Command and Control Schema, Revised Draft, Version 0.5.3. Unpublished report.
- Cohen, Paul R. (1998). Dynamic Maps as Representations of Verbs. Submitted to ECAI-98.
- Jones, E., (1998), "Formalized Intelligence Report Ensemble (FIRE) & Intelligence Stream Encoding (ISE)", Alphatech memo.
- Lee et al, (1996) The PIF Process Interchange and Framework Version 1.1, Jintae Lee (editor) Micheal Grunninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost and other members of the PIF Working Group, published as MIT Center for Coordination Science, Working Paper #194, 1996. Available via <http://soa.cba.hawaii.edu/pif/>
- Lenat, D. and Guha R., (1990) Building Large Knowledge Based Systems. Reading, Massachusetts: Addison Wesley.
- Lenat, D. (1995) Artificial Intelligence. *Scientific American*, September.
- Park, J., Gennari, J., & Musen, M., (1997). Mappings for Reuse in Knowledge-based Systems. Stanford Section on Medical Informatics technical report SMI-97-0697.
- Pease, A., & Albericci, D., (1998). The Warplan, Teknowledge, Palo Alto, CA, March 13.

Pease, A. & Carrico, T. (1997). "The JTF ATD Core Plan Representation: Request for Comment". Armstrong Lab: AL/HR-TP-96-9631.

Pease, A. & Carrico, T. (1997). "The JTF ATD Core Plan Representation: A Progress Report", Proceedings of the AAAI 1997 Spring Symposium on Ontological Engineering.

Srinivas, Y. & Jullig, R., (1995) Specware(TM): Formal Support for Composing Software, Proceedings of the Conference on Mathematics of Program Construction, Kloster Irsee, Germany, July.

Tate, A. (1998), "Roots of SPAR - Shared Planning and Activity Representation", to appear in Knowledge Engineering Review, Special Issue on Ontologies, Cambridge University Press.

Wiederhold, G. (ed.) (1996): Intelligent Integration of Information; Kluwer Academic Publishers, Boston MA, July 1996.

References

- Cohen, P., Chaudhri, V., Pease A., and Schrag, R. (1999), [Does Prior Knowledge Facilitate the Development of Knowledge Based Systems](#), In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-1999)*. Menlo Park, Calif.: AAAI Press.
- Cohen, P., Schrag, R., Jones, E., Pease, A., Lin, A., Starr, B., Gunning, D., and Burke, M. (1998), [The DARPA High Performance Knowledge Bases Project](#), AI Magazine, Vol. 19 No.4, Winter.
- Ford, L. and Fulkerson. D., (1956). Maximal Flow Through a Network. *Canadian Journal of Mathematics*, 8:399-404.
- Li, J., Condoravdi, C., and Pease, A., (2000), [From Visual to Logical Representation: A GIS-Based Sketching Tool for Reasoning about Plans](#), Teknowledge Technical report, January 9.
- Pease, A., Chaudhri, V., Lehmann, F., and Farquhar, A., (2000), [Practical Knowledge Representation and the DARPA High Performance Knowledge Bases Project](#), In A. Cohn, F. Giunchiglia, and B. Selman, editors, *KR-2000: Proceedings of the Conference on Knowledge Representation and Reasoning*. Breckenridge, CO, USA, 12-15 April 2000, San Mateo, CA, 2000. Morgan Kaufmann.
- Pease, A., Liuzzi, R., & Gunning, D., (2001), [Knowledge Bases](#), in *Encyclopedia of Software Engineering, Second Edition*, ed. J Marciniak, Wiley & Sons, NY.
- Pease, A., Niles, I., (2002), IEEE Standard Upper Ontology: A Progress Report, Knowledge Engineering Review, to appear.